DEFINING A VIABLE MULTIMEDIA COURSEWARE ENGINEERING APPROACH

Educational Research by Michael Shaw

December, 2000

CONTENTS

ABSTRACT

Multimedia courseware that is well designed can facilitate and promote very positive and deep learning experiences for end users. There are many issues surrounding the development and use of effective educational software systems that will produce these types of meaningful learning activities. One of the biggest problems encountered is the disjointed relationship between what a client wants or expects and what is in fact realistic. Related to this are factors such as cost and quality. By examining various attributes of common courseware products and their associated engineering concepts, tools and practices, we can better understand how to put engineering systems in place through examples and best practices that will increase efficiency, effectiveness and accuracy, and reduce time to delivery ratios. This paper discusses the product and process, and various management strategies. I use the term 'approach' in the title so as not to raise expectations about establishing a rigid framework for the industry as a whole.

INTRODUCTION

Beyond earlier and more traditional CBT platforms, the Internet itself can be classified as a multimedia distribution system, and its associated technologies are being rapidly developed to provide faster, friendlier, more interactive, and more powerful means of developing and distributing courseware – whether it be to a campus, home and/or the workplace. In many instances this can be an over-simplification, and the information from much of the literature I've read, and many of the people I've spoken with, would seem to indicate that many individuals and organizations have adapted their own unique approach to courseware engineering. Many of these appear to be borderline dysfunctional, and lack standards and quality control. Although defining and implementing viable practices and procedures can be very particular to organizations or individuals, there are a number of commonalities, including basic models, practices and popular tools, that can be instrumental in developing and producing good courseware in an efficient and cost effective manner across all delivery platforms.

The size of the organization and the expertise available appear to be some of the most critical factors leading to a wide range of approaches. Further investigation indicates a host of more subtle contributing factors, ranging from the domain content to an institution's vision for teaching and learning. Many courseware projects may be intuitively designed, but consequently ill defined and structured, difficult to standardize, and prone to failure. Clients are not always pleased with results. In order to put an effective strategy or system in place, it is important to first define and understand the mechanics of multimedia courseware engineering and investigate how to apply appropriate knowledge and tools in a variety of situations to yield quality products.

DEFINITIONS AND REQUIREMENTS

The software engineering field is somewhat similar and related to courseware engineering, in that it may possibly provide us with structured processes around fundamental development issues. Some of the design aspects - such as the user interface, and the production aspects - such as coding, can be similar as the medium and the tools may be similar. However, in education we require more emphasis and attention on the end user or learner, so that learning outcomes are effectively and efficiently met. A clear emphasis on human learning sets educational software apart from software in general (Bell, 1993). Courseware engineering may embody many of the developmental aspects of software engineering, but it is more specific to the development and implementation of educational software systems that support teaching and learning activities. This can include a whole spectrum of activity, including defining the learning needs or problems and all of the relevant tasks, agents, products, tools, contexts, resources and costs involved. MCE embodies the very essence of educational psychology, analysis, design, methodology and evaluation. This prompts us to pay more critical attention to instructional design issues and all of the associated processes, which transcend software engineering's methods for developing effective and efficient software and development processes. In this context, courseware engineering is much more encompassing than software engineering, as it also includes all of the instructional theory and its practices, tools and methodologies required for developing and delivering quality multimedia-based educational experiences. Further, little attention has been given to the technical foundations of educational software. Areas such as reliability metrics or product complexity analysis are not commonly related (or viable) to educational courseware engineering (De Diana and van Schaik, 1993).

A courseware engineer could be quite a multi-faceted character if in fact a single person is relied upon to develop and produce all of the components necessary to create an effective electronic learning product. Educational multimedia developers can come from many areas, including the domains of art, literacy, film and television - being programmers, graphic designers, artists, animators, musicians and instructional designers. This is because the media incorporated into courseware can be complex and may require specialist skills in the production of photo-realistic images, animations, audio and video, and have special programming requirements. In respect to this integration, each such medium needs selecting, designing and producing. The development, production and integration of these components and other related activities can significantly impact on the delivery time and resources involved. In addition, coordinating and monitoring all of the resources and complex activities is no small feat. What is evident here is that the concepts of teamwork and multidisciplinary approaches, and project coordination and management, are necessary to bring substantial multimedia courseware projects to fruition (England & Finney, 1999). Perhaps the best vocational designation for a courseware engineer would be that of 'project manager'.

Clearly we are dealing with a process and a product. Beyond simply realizing the potential inherent superiority of computer-based delivery platforms, well-

orchestrated design strategies that are appropriate to the technology and a well-defined process to formulate these strategies into a viable product is what is required. These days with the nascent proliferation of computer-based learning systems, I can really see the need for a more structured, coordinated approach for developing courseware - one that, in particular, includes defining and implementing successful components that are re-usable, whether they are part of the product or the process. Recurrent practices can embody much expertise, but this expertise can be difficult to translate into a methodological language for systematic replication (Goodyear, 1995). To a lesser extent, we must also consider the platform intended for delivery, which has its own issues, and can determine the methods, tools and practices employed.

I believe that by separating, but not de-coupling, courseware engineering issues into respective product-based and process-based categories, we can begin to develop a clearer picture, and formulate a more organized approach to creating better development systems, and consequently, improved end products.

PRODUCT AND PROCESS-BASED ISSUES

There can be a huge difference in how we view a product based on its scope. It could be a large complex system created by teams, a desktop courseware project developed by an instructor or a by-product of a learner's activities (Mayes, 1993 in Goodyear, 1995). In this paper we will assume that the type of product in question will be of substantial scope, either being a complex system or one having a significant impact on a system. From here it is possible to scale down our ideas so that they are more easily adaptable to smaller products.

I believe that most departments, companies, institutions and individuals have learned to consistently develop a product in their own unique style. This style is usually evident in the final product. Even with this, what a product often actually ends up being is quite different from what it should or could have been. The expectations of stakeholders and the reality of what is possible need to be clearly defined before any process begins (England & Finney, 1999). Everyone involved in a courseware project can become so enveloped in the process that they lose sight of what they are supposed to be developing. Clients can focus their requirements if they can see or use something that resembles the final product. Prototyping may not always be a first or viable option for providing clients with this type of example.

In hopes of creating an accurate specification for educational courseware in general, we consider here some of the common attributes of a quality multimedia courseware product. These attributes can lead to quality standards that become embedded into a process, rather than left to chance of becoming discovered in one. Some may rightly argue that this type of definition is in fact part of the overall process in determining what the initial specification is. The idea here is simply to create an arbitrary division for the purposes of determining what stakeholders and the industry will generally regard as a quality product.

I investigated a recent Canadian study on quality initiatives in courseware, which determined that a number of factors needed to be present in a quality, learning product (Survey of Perceptions, 1998). My gist is that many successful educational multimedia courseware products exist, and most share common traits. To summarize the report in the context of this paper, I have drawn on the responses of those surveyed - being experienced educators, developers and producers.

- As we are dealing with an educational product, pedagogical integrity is of paramount importance, and includes many factors. It should embody learner-centred design principles that engage the learner, and incorporate instructional methods that are well matched to the type of technology used. In other words, the methods should enable or trigger learning mechanisms consistent with new learning paradigms.

- The product should be suited for its intended audience, but courseware that is flexible enough to be used in a variety of instructional settings will have a broader appeal and may be more viable and/or marketable.

- The product's learning objectives should be clearly stated, so that its usefulness can be easily determined, especially if it is to be marketed.

- Core competencies, performance standards, learning objectives, learning activities and assessment activities for learners should be accessible through an easy-to-use, intuitive interface.

- Audio-visual design principles are also important. They should be consistent with the user interface, and there should be balance between the various media and their selected components that are incorporated throughout the program.

- Compatibility with handicapped devices is a big issue in Canada these days. For example, streaming audio or video files with closed-captioning would include those learners with hearing impairments. A conforming product could get a big government seal of approval and be more marketable.

- The mechanisms that test and assess student achievement should conform to quality standards. There is also talk in Canada of standardizing student management and tracking facilities.

- Content must be timely, accurate and credible, and must not infringe upon copyright law.

More research may be required to further identify issues of product quality for specific stakeholders, but this short summary provides us with an indication of what the educational community in general considers to be critical to multimedia courseware quality. With this, we might attempt to construct a process to develop

and produce a product that will better meet the quality standards and needs of the stakeholders.

Aside from poor product understanding between stakeholders and development teams, two other process-based factors to consider with multimedia courseware engineering deal with the competencies of the development team and the actual phases of development. Again, there are a number of viable approaches, but we can set out to establish some overall guidelines.

I have observed that team expertise and the execution of development phases and steps are fundamentally related. In one particular instance I recall vividly, a team of novice developers attempted to work through a project according to a predetermined process schedule set out by a software program known as *Designer's Edge*. The project ended up becoming a process that was in perpetual design. This was due to the fact that each new discovery in the group led to new insights and understanding, which prompted revisiting each challenge, and each phase in the process. Thus, without well-defined criteria for feedback, the group became stuck in a perpetual loop - only escapable at a time when expertise and awareness became developed enough to allow them to progress without constantly reworking each step. One could blame the incompetence of the project manager, who should have realized that additional resources were required for the project, and who gave too much free reign to novice participants on certain issues. Regardless, the situation created a valuable learning experience for the entire department.

Another way of addressing process issues is through the use or creation of knowledge rich tools that support courseware design. There is a proliferation of courseware development tools in use today, and it is estimated that there are over 100 products on the market in North America alone. Rather than providing guidance on how to use technology to create meaningful learning experiences, popular tools such as WebCT and Blackboard typically do nothing more than act as a hypermedia filing cabinet for whatever people put in them, and offer a few communication and other basic tools. On the micro level we consider the product developed by the average tutor that incorporates these tools into their practices, or even the learner developing material (sometimes referred to as knowledge engineering). On the macro level we can consider developing the knowledge rich tool itself, which could be designed to meet the particular authoring or even learning management needs of a group or organization.

Bell (1998) investigated a simple tool that incorporates solid instructional theory with guided case adaptations together with an easy to use interface. This is the type of tool that is desperately needed for ubiquitous technology dependant learning experiences today. It conforms to the idea of producing a quality product through a re-usable quality process. This particular example employs an instructional design theory known as Case-Based Reasoning (Schank, et al. in Reigeluth, 1999), which suggests that learners should learn "how to" rather than "know that". It incorporates goal-based scenarios, which invoke higher order learning that transcends knowledge and comprehension. Goal-based scenarios provide a learn-by-doing situation in which students incorporate personal

significance with knowledge and apply it. In other words, the content and skills learned are necessary to achieve goals that are interesting and important to the learner. Case-based reasoning deals with how we remember and use our memories to solve new problems. In essence, goals that learners find interesting, important or even fun can through practice help develop skills that stay in long-term memory. This creates a founding information base that will eventually turn novices into experts. Schank also suggests that curriculum be changed to reflect three primary emphases -- communications, human relations and reasoning - the three types of knowledge competencies required to succeed in today's society.

The specific product that Bell's Investigate and Decide Learning Environment (IDLE) tool helps design novices to produce may be a little too procedural and very limited in allowing learners to address true heuristic investigations, especially in ill-structured domains. Another example is the Experimental Advanced Instructional Design Advisor (XAIDA) tool (Hsieh et. al, 1999). These types of tools are typically designed so that they are more domain-specific. For example, Chemistry 101 templates could have selectable, standardized molecular models already built into the software.

What we can learn from these examples is that truly effective development tools should incorporate pedagogies based on the principles of active learning, multiple perspectives, collaboration and knowledge building, not only in accordance with the subject matter, but also with the powerful features of the particular learning technologies in mind. From this we can search for common design parameters and apply them to develop a viable tool for a specific domain. Some of the key instructional themes, which are in line with what a quality educational product and process should incorporate, are:

- Problem solving
- Initiative and self direction
- Learner or user centered environments
- Real-world situations
- Hands-on, learn-by-doing approaches
- Active building of knowledge
- Personal involvement
- Peer collaboration
- Time to persevere
- Guidance and support
- Incorporating advanced technology

On the macro level, I think about how such a development tool could have helped the team stuck in a process-loop in my earlier example. The whole concept of these types of tools can provide some relief in resolving process-based issues, but at their current level of development, it would most likely be limited to products of a small scope. Ideally, a quality tool would help select an appropriate instructional method or theory to match learning objectives with audiences and domains.

There are a number of other types of software tools available to automate different aspects of courseware development. Strong automation suggests the actual replacement of human activity. Strong automation tools would automate major design decisions, but their structure may severely limit a more needed empirical approach to design. Weak or passive automation would refer more to forms of support for the human agent, who is in control (Goodyear, 1994). Weak or passive automation tools could include support for project management, analysis and design. Throughout this paper, we will examine a variety of these tools.

Not every educational institution is fortunate enough to have a skilled team of courseware engineering professionals on staff. Certain aspects of a project may have to be farmed out to external sources, or because of budget restrictions, developed in-house through strenuous efforts. It is evident that, in a project of substantial scope, the expertise and skills available must be structured properly and guided through the phases of a viable process in order to produce a quality product. This illustrates the critical need for project management principles that can unite the disparate ways of working in interactive multimedia development (England & Finney, 1999). Most importantly, this also suggests that automation of the process can provide various levels of support where needed.

DEVELOPMENTAL MODELS FOR MCE

Obviously, there are a number of practical goals to be considered specific to courseware engineering that require a problem-solving process. These goals typically relate to the tasks, agents, products, tools, contexts, performance indicators and costs involved (De Diana and van Schaik, 1993). In any complex project environment, developmental models are required for various reasons. They can provide developers with different levels of abstraction in a framework in which to solve problems. Linear, sequential development is not necessary, but initially a sequential framework can provide a more organized way of viewing the different stages required to bring a project to completion. There are many models and variations available for developing courseware. As we will see, they are basically variations on analysis, design, production, evaluation, revision and management activities.

Project managers will need to monitor and control progress, and a model can provide an opportunity to help identify and target specific goals throughout each phase of the process. Clients may be required to sign-off at particular stages. This is sometimes referred to as a milestone approach. A well-designed model provides for a more coordinated way to manage all of the project’s activities, including establishing time frames, and cost estimation and control, and can help to ensure that quality standards are being met.

The problem with most lifecycle process models is that they can be oversimplified and therefore poorly suited to a particular courseware project or project team. This was found to be especially true in the case of software development models that do not support more effective approaches for prototyping and software reuse,

and therefore increase a number of risks (Boehm, 1988). If we are to borrow from the more or less structured processes of software development, then we to must consider the need for models that support concepts such as prototyping, and activities and products that can be sequenced or repeated. Recursion to previous steps in the development process is a strength, and generally speaking, the more evaluation-revision cycles there are, the better the product. However, these activities have to be organized and orchestrated in a way that does not create a perpetual design cycle that is both time consuming and costly (Figure 1).

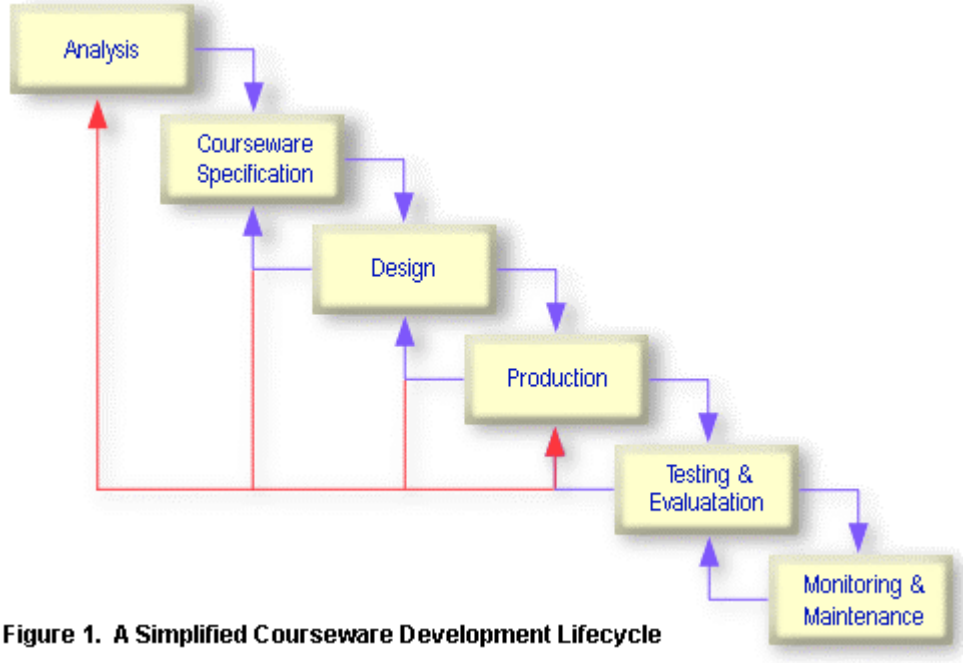


Figure 1. A Simplified Courseware Development Lifecycle

This is but one example of a quasi-waterfall model for courseware development. In true practice, the linear sequence suggested is far less ordered. The arrows indicate potential feedback for exploratory, experimental and evolutionary development and/or prototyping, which is all very non-specific. It could represent the predicament of a development team that doesn't know when to stop prototyping or experimenting and actually start developing.

Even with refinements, the waterfall model in software engineering has had fundamental difficulties. In interactive end-user applications (those of which we are most concerned), the production of spurious documents as completion criteria for early design phases (milestone approach) has led to poorly understood concepts, and hours of lost work because of unusable code (Boehm, 1988). The client may also have trouble interpreting the sign-off documents containing the requirements analysis, specification or design detail, or may not even understand their own requirement at all (Goodyear, 1994).

The early development stages of analysis and design in multimedia courseware engineering are critical and much different than those found in software engineering. This is where a systematic approach to designing instruction and instructional materials to achieve specified learning objectives takes place. This involves identifying the appropriate learning methodologies as well as analysis of

the subject matter and/or skills to be learned. However, the production, testing and maintenance stages are similar in both courseware and software engineering (Bell, 1993). It is important to note here that instruction design has its own unique processes and methodologies for solving problems and defining and refining goals. Becoming embedded in a courseware engineering process though, suggests that instructional design should have a greater focus on creating an understandable, consistent behaviour within the entire instructional system. Instructional design activity should centre on making predictions about learning processes and decisions concerning the instructional activities within the whole system, rather than attempting to manipulate learners (Goodyear, 1993).

There are certain characteristics of technologically rich learning environments that must be considered when developing instruction. Integrating multimedia courseware with good teaching practice, learning theory and instructional design can develop the types of critical thinking skills associated with more effective, higher-order learning. Multimedia is a vehicle, which can incorporate multi-modal representations that appeal to dominant and auxiliary learning modalities in learners, and can help foster things such as collaboration and flexible cognition. Keeping this in mind, we consider the analysis phase, which feeds into and provides the foundation for all subsequent courseware design and development activities. Analysis typically involves the following:

- Identifying the (instructional) problem and source
- Identifying instructional needs
- Determining entry skill levels
- Setting instructional goals

The following should also be taken into consideration at this point:

- Duties and/or performance expected from users (task analysis)
- Parameters for feasible performance measures (validation)
- Scope and depth of the content material
- Delivery environment, platform(s) and other technical needs

Analysis begins with (re) defining the problem, identifying the source and determining possible solutions. Information is required. Developers and stakeholders should make themselves aware of any viable information already available in their organization before conducting specific research activities.

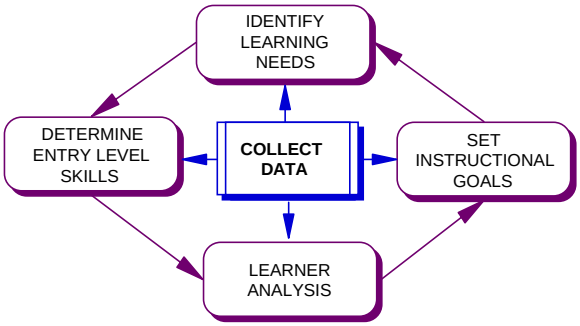


Figure 2. Data Collection and Activities in the Analysis Phase

Standard software packages can serve as useful tools in the collection of information. For example, form design packages can help with the collection of data, and spreadsheets and statistical packages are useful for analyzing, summarizing and charting data. Various other methods such as videotaped interviews and demonstrations can serve as excellent information resources.

There are many ways to determine what skills, knowledge and/or attitudes the target audience needs to acquire. A basic discrepancy approach can be used which contrasts the existing skill norm within a group with the final skills desired. The difference represents the required learning and helps determine the learner's entry level. These needs may then need to prioritized and analyzed for relevancy to the project, keeping in mind that we are training people, and not simply improving the tools that they are using.

It is important that user entry levels also take affective, social or political issues into consideration as these factors can greatly influence learner engagement. We might also consider what types of technical skills the learner's will require to engage the courseware. This may require the development of a special module that introduces learners to the proper use of the software or system.

Learner analysis involves finding out all you can about your target audience. There are various ways of collecting information concerning the learners, including ethnographic surveys and observation. Obtaining these data should consider the following factors, which may or may not be interdependent:

- Who they are
- Where they live
- What their attitudes are
- Their age(s)
- Levels of education
- Cultural backgrounds
- Past experiences
- Special needs
- Literacy levels
- Preferences to learning modalities

A set of initial goal statements can be corroborated with the defined needs and audience analysis data. Typically these goals are a brief description of an instructional intent and are quite broad in nature. Later, performance objectives will be developed from these goals that will ensure that the instruction is measurable, achievable and technically viable. Tasks will also be developed with specific instructions on what the leaner has to do in order to achieve the goals and objectives. Performance objectives describe what the learner should be able to do after the planned instruction is carried out.

Software programs that help generate graphical representations (Figure 2.2) are real time savers. For example, they can allow the visualization of the learner's existing and required skill sets and bodies of knowledge or representations of the subject in a semantic net, a hierarchy of concepts or a flowchart of procedures, which will vary depending on whether the content is based on cognitive skills, verbal information, cognitive strategies, attitudes or motor skills. This type of visualization tool can prove very useful when creating tasks and objectives. Using the same software program, these can easily be incorporated later into navigational charts, which show how the learner will move through the instruction and the multimedia environment. Depending on the circumstances, these tools may prove very useful in communicating with stakeholders or other developers.

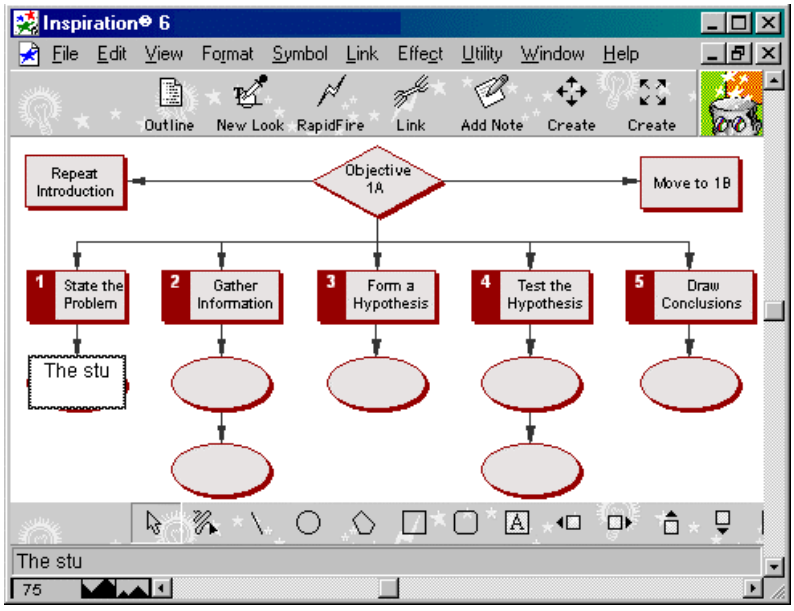


Figure 2.2. – Inspiration Screen

Inspiration is an example of a software tool that quickly and easily creates graphical representations that are ideal for visualizing various courseware engineering components in the analysis and design phases.

The translation of analysed needs into designed functions can be difficult. By properly matching the characteristics of specific technologies to learning, we can help ensure that the analysis phase is not constantly revisited. Each type of delivery system may organize content resources and elements such as exercises, problems, and feedback in various ways. The advantages and limitations of the system to be used should be taken into consideration while developing the goals

and task analysis. For example, bandwidth limitations could impose serious restrictions on the types of media used, and greatly influence how the course is delivered and it's level of interactivity.

With a set of existing skills and/or sub skills defined, we can begin think about performance/instructional objectives for each goal, and begin to verify that they are in fact be achievable and measurable. The conversion of the learning goal analysis into content analysis begins the design phase. This is usually an overlapping area, as the way in which the tasks are performed and the goals and objectives met can be dependent on design factors. We will also see that in the design phase, some scheme for classification of instructional objectives into effective types of strategies is required, and these may affect the way in which we describe goals and tasks.

The design phase takes its input from the descriptions created in the analysis phase (often referred to as the specification), and will output a detailed description so that we can begin to see how the program will actually be physically created. It will provide feedback for evaluation and revision of previous activities.

The design phase consists of the following components:

- **Instructional design theory**
  - appropriate selection of instructional design theories (derived from specifications in the analysis phase)
  - instructional objectives

- **Learning environment**
  - selecting and ensuring that the necessary components are in place

- **Content design**
  - content scope and outline or treatment (from task analysis and/or goals)
  - media determinations (linked to objectives)
  - storyboard and/or script(s) for multimedia or media components

- **Presentational design**
  - interfaces, navigational elements, layout, look and feel
  - artistic elements – graphic design and/or other media design
  - template or prototype design

A software program that can help with analysis and design activities is Designer's Edge (Figure 5.1), which is briefly discussed later. It incorporates tools for all phases, including task and needs analysis and design.

We can see that there is much activity in the analysis phase leading to the design phase, and a tremendous opportunity for unpredictability and endless loops exists. Therefore, the initial requirements should be specified clearly so that analysis-design phases are not part of an endless evaluation-revision cycle. There are various ways to deal with this. For example, instead of working through all of the instructional activities at once, we can prioritize them into a lesson or module that

will best represent the entire system to be developed. By illustrating the principles and usability of a planned system through a prototype, we can help to clarify the initial specification of requirements. This is where we can employ the general quality product guidelines set out earlier in this paper, to help create an initial visualization prototype, which will help establish the perceptions and expectations of the client (Figure 3). This reduces the risk of prematurely developing several prototypes of a system, only to find out that a number of errors have been made in the initial requirements phase. Diverse modules may require multiple prototypes and cost calculations.

Authoring tools can be used not only to prototype, but also to actually refine and produce the finished courseware. These will be discussed in more detail later.

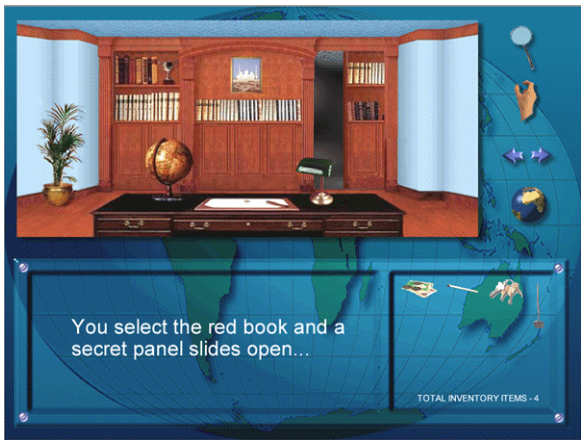


Figure 3. A screen from an initial prototype for an educational game developed in Toolbook. There is basic navigation and functionality, but the majority of the instructional content is absent from the storyline and sequenced events.

Stakeholders were required to sign-off on the interface, presentational design, basic storyline and basic instructional approach.

Michael Shaw, HCT (1997)

We could even consider this approach in the modular design of courseware, in that the first module is the initial prototype, and the only one that needs intensive revision, as the subsequent modules will be all based on it. Beyond this initial prototype, a rapid prototyping scheme can then be employed for efficiently developing more specific aspects of the project. Timely examples, through carefully orchestrated specify-design-implement-review loops, can provide all parties involved, especially the client, with a better understanding of what the product will end up becoming, and increase the probability of success (Goodyear, 1994). With this improved focus, formative evaluation and revision can be applied more efficiently throughout the development process. The proliferation of multimedia authoring tools these days makes rapid prototyping much faster and easier than ever before.

Rapid prototyping in educational software development has its own unique challenges, as the project's learning objectives may not be realized or met in any partial test scenario. Learner's opinions may be valid in the development process, but judgements about the overall effectiveness of learning may have to wait until a more complete version, module or lesson is ready for actual field-testing. Therefore, in developing a viable courseware engineering model and approach, we must make critical analysis and instructional needs a priority, in order to minimize the risks of failure or major re-works.

Over the years, refinements on the waterfall model have led to interesting rapid prototyping models. The spiral model (Figure 4) suggests a development lifecycle as a number of repeating cycles with a similar sequence of steps. This could be applied at many levels – from a product's overall concepts of operation down to each portion and level of refinement.

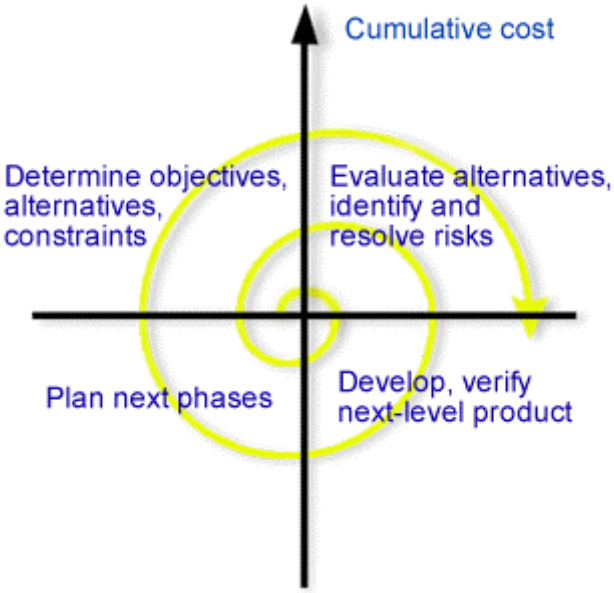


Figure 4. A Spiral Model of the Software Engineering Process
(Adapted from Boehm, 1988)

In the Boehm spiral model, the spiral begins with a particular task that could be improved upon. By determining the objectives of this component to be elaborated upon, and by considering alternative means and the associated constrains of implementing it, we can evaluate areas that present sources of project risk. Higher-risk components are given priority. Assessment of these risks determines the next refinement in the loop or spiral. Refinement makes the selected component more specific. Compliance testing ensures that all project participants are aware of the progress and are prepared to plan for the next cycle in the process (Boehm, 1988 & Goodyear, 1994).

Ideally the spiral would terminate with the installation of the product, but many external factors (i.e. - time, cost, new discoveries) may cause it to terminate earlier. This type of model is very flexible and it can support milestone or prototyping approaches or a combination of both.

There have been many courseware development and engineering models published. The theories of instruction that underpin them are similar, and they usually differ because of the different environments and situations they are used in. It could also be that instructional theories were not thoroughly considered when the models were formulated. Universal solutions may be unrealistic, but by considering the different steps found in various courseware development models, we can conclude that there are several common functionalities from which we can draw on:

- Analysis
- Design
- Production
- Evaluation
- Refinement and revision
- Process management and control

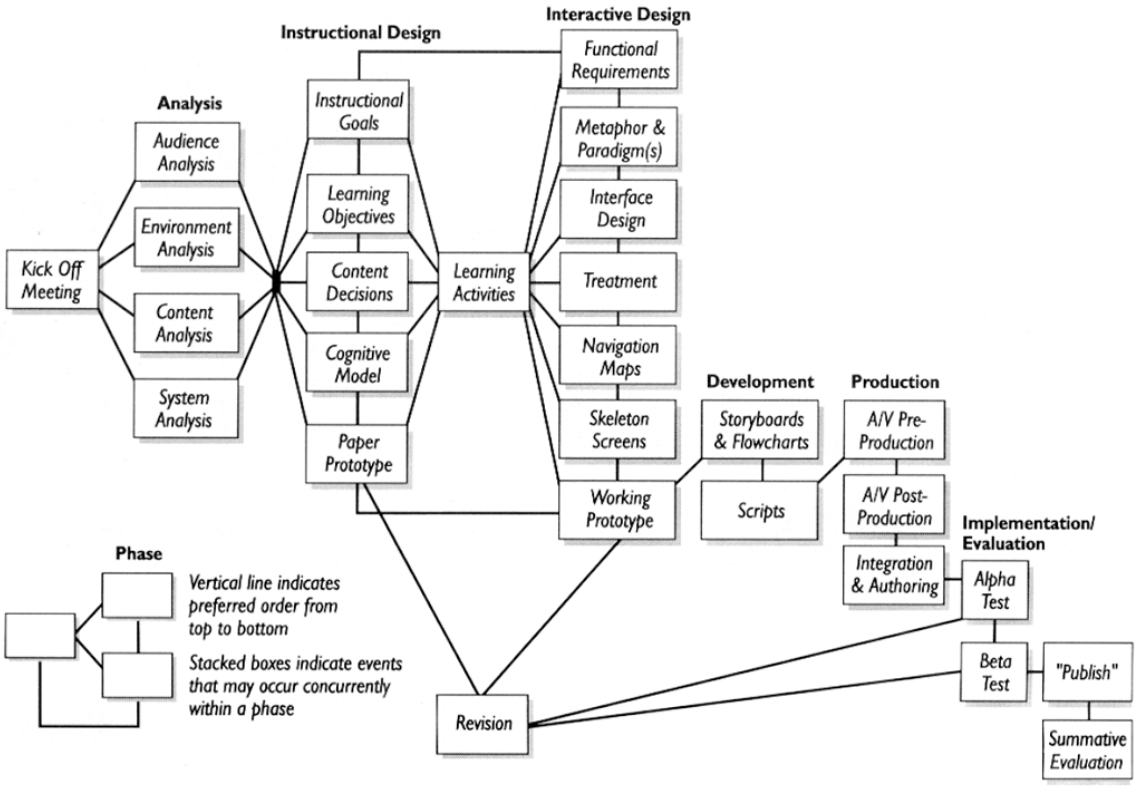


Figure 5. Multimedia Design and Development Methodology
(Brian Blum, 1993 in: Multimedia: Making it Work, 1994)

As mentioned earlier in this paper, many organizations and institutions have developed their own methods of development over time, based on experience and best practices. Figure 5 is a rather impressive looking model, which represents the type of detailed process that can be established by a particular faction. In this example, all of the common functionalities are present, and there are many loop-backs for revisions based on testing. However, there is no real defined prototyping scheme. The production of each medium and related software control could proceed in sequence rather than in parallel, and the process could get quite complex and overwhelming. What is important to note here is that the way this process is perceived may in fact be quite different from how it is actually used in real life situations by its creators.

Many factors can contribute to the way in which a process is organized and orchestrated, especially the size of the development team, and the level of skills

and knowledge available. The focus of course should always be on the quality of the end product and the quality of its integration into a learner's course of study.

I have included examples of published courseware development and engineering models here (Tables 1 & 2) as a basis for comparison with the ill-defined, high-level abstraction lifecycle model in Figure 1.

Analysis	• Define training requirements • Analyse target populations • Establish performance levels
Design	• Specify instructional objectives • Group and sequence objectives • Design instructional treatments
Production	• Develop learning activities • Develop test items • Evaluate prototypes
Implementation	• Implement learning activities • Administer test items • Assess student results
Maintenance	• Revise course materials • Revise test items • Assess course effectiveness

Table 1. A Practical Courseware Development Model
(Adapted from Spector et al.,1992 in Intelligent Frameworks for Instructional Design)

The above model is specific to courseware development. In it, we see a concentration on the types of activities directly relating to learning, but the process is rather simplistic and still not refined enough to offer a detailed understanding of the complete process.

PHASE	GOAL	INPUTS	TASKS	OUTPUT
Feasibility study	Green flag for go ahead	Client with requirement	Low detail in: Target population description Task analysis Domain analysis Develop alternative implementation strategies Constraint analysis Cost analysis	Feasibility report with costed-out alternatives
Needs	Describe current	Initial	Describe target	Requirements

Analysis	situation, learning points/objectives	requirements from feasibility	population (knowledge, motivation, ability, variation) Training needs analysis (learning objectives hierarchy) Learning needs analysis (subject domain) Describe environmental constraints (access, hardware, operating systems...) Develop evaluation documentation	specification, a theory of learning/instructional framework
Design Phase 1	Production of media requirements including software requirements	Requirement specification, instructional framework	Conceptual/instructional design: Pedagogical design Generic media selection Content/curriculum structuring Instructional events and sequence Presentational design: Script for each medium (including control software) Navigation design User interface design Peer/expert review	Functional specification
Production Phase 1	First working prototype	Functional specification	Development/recording of media according to scripts Editing of media Integration of media	Media master copies, first working prototype
Prototyping Cycle	Software for pilot testing	**Initial, then successive prototypes**	Quality review Evaluation of alpha test version for robustness, usability by peer review as necessary: revision of design, revision of software	Working software for pilot (beta) test
Pilot and Field Testing	Evaluation of beta test version	Beta test software	Evaluate using pilot or/and target (filed) group	Finished courseware ready to install

			As required, revise design, revise software	
Installation (including any documentation for users and maintenance)				

Table 2. A Practical Courseware Engineering Model
(adapted from Bostock & Drummond 1995, modified by Bostock in 1998)

This more concise engineering model gives a better overall understanding of the process involved, including creating initial, then successive prototypes.

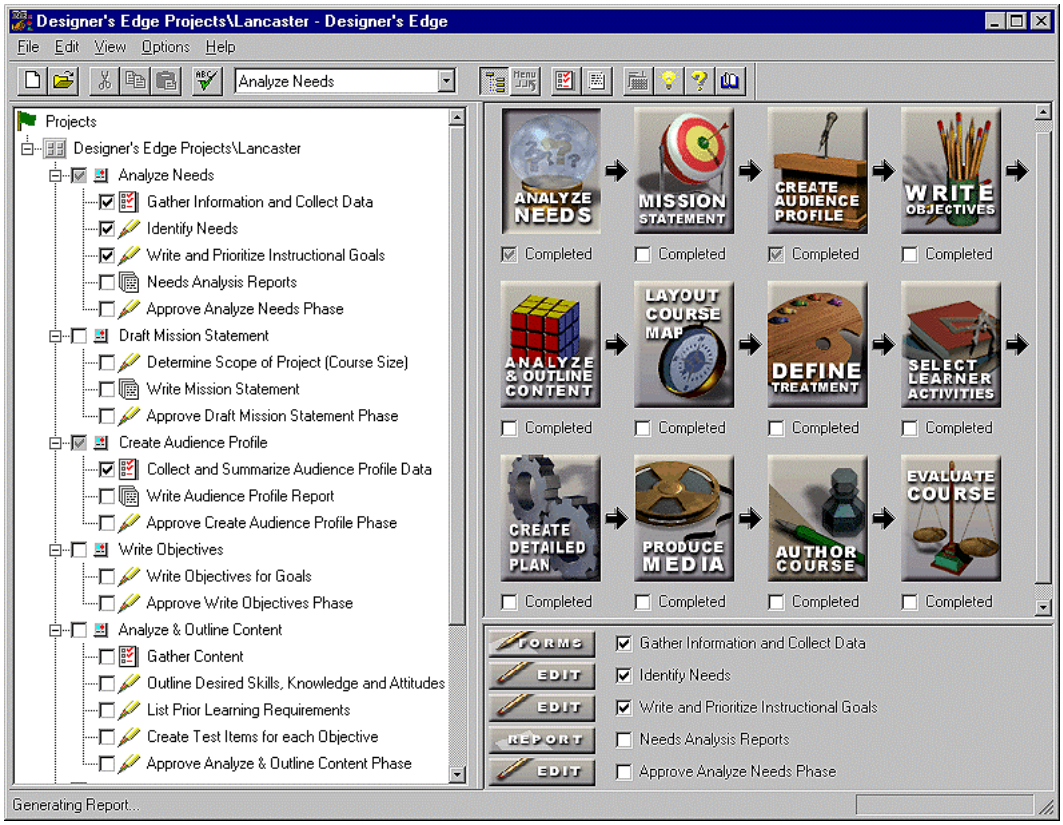
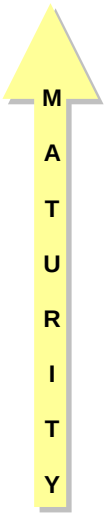


Figure 5.1 Allen Communication's Designer's Edge Program

Various software tools can help to organize and automate the whole courseware development process. Designer's Edge (Figure 5.1) takes the traditional instructional design model of analysis, design, development, implementation and evaluation, and sub-divides it into distinct units. The interface can be customized to reflect the specific needs or standards of a particular project. The program also generates a number of forms, storyboards and other printed documents. Storyboard tools create frame mock-ups complete with content, interactions and media selections. A visual course map is constructed as the course is developed. This tool in itself provides a structured model for courseware design. As with most other support packages that aid in the whole development process,

Designer's Edge must be used faithfully and understood by all parties involved in order to be effective.
In searching for a viable MCE approach, we must realize that the processes we follow are fundamentally related to the principles we apply. These methodologies are not always initially understood or fully developed, but they can become structured and recurrent through maturity in a particular working environment. Figure 6 is based on a corporate maturity framework, and is relevant to infrastructure development or evolution in courseware engineering. In the early or beginning stages we have an ad-hoc process with little or no formal procedures. At the high end of maturity we have a process, which is not only proven, but is actually self-improving. Recurrent practices and officially sanctioned methods and practices can be viewed as structuring resources. Realizing that there may in fact be a maturity process at work can help various organizations and departments plan for their own evolution (Goodyear, 1994). With this, we might consider not only how an organization is 'growing-up', but also the whole courseware industry in general.



OPTIMIZING – A self-improving production process is in place: data is routinely gathered and used to identify problems and new opportunities for improvement

MANAGED – Quality and production measures are established and recorded: a quantitative production process is established

DEFINED – Qualitative processes are established: a system with the potential for improvements is created

REPEATABLE – There are some controls and re-use of experience, but no framework for improvement is in place: process may be dependent on key personnel

INITIAL – An ad-hoc process with no formal procedures, including monitoring, planning and tool integration: management not aware of issues

Figure 6. A Corporate Maturity Framework
(Adapted from Carnegie Mellon U. in Goodyear, 1994)

Developmental models can provide us with a framework in which to solve problems, and can assist us in developing a system of viable methods and principles. We can tightly structure a bridge from abstraction to concrete solutions that produce successful results by adapting MCE models to our real-life working environment.

With an understanding of a MCE framework in place, we can begin to consider dividing the phases down even further and organizing steps into more manageable tasks. An outline and plan must consider and manage the skills, time, budget, tools and resources that are available to bring a quality multimedia courseware product to fruition.

DEVELOPMENT TOOLS FOR TECHNOLOGY BASED LEARNING

No courseware engineering approach would be complete without discussing a few of the popular authoring tools available today that support technology based learning. It's important to differentiate between authoring systems and learning management systems. Authoring systems enable the production of interactive multimedia courseware. These typically incorporate support for a range of question types and response analysis. Learning management systems (LMS) are software packages that support the management of learning in an organization. LMS's can include support from small stand-alone classroom applications to those that will be distributed over a LAN, WAN or the Internet and intranets (Guides for Managers, 2000).

The wide proliferation of these tools has enabled many people to become more actively involved in creating multimedia-based instruction. Earlier versions typically had a high learning curve, but the fact is that they are becoming easier to use all the time. Content experts and other non-programmers are now authors, and authoring and LMS tools are being used on a routine, daily basis by many tutors in many institutions.

Based on my own workplace experiences, I will examine a few popular authoring tools and a typical learning management tool, and discuss some of their practicalities and shortcomings. The first four software tools are authoring tools that were primarily developed for CD-ROM based multimedia applications, but have evolved with the ability to create web pages or be streamed over a network, intranet or the Internet. The LMS tool I have selected is WebCT.

Authoring Tools

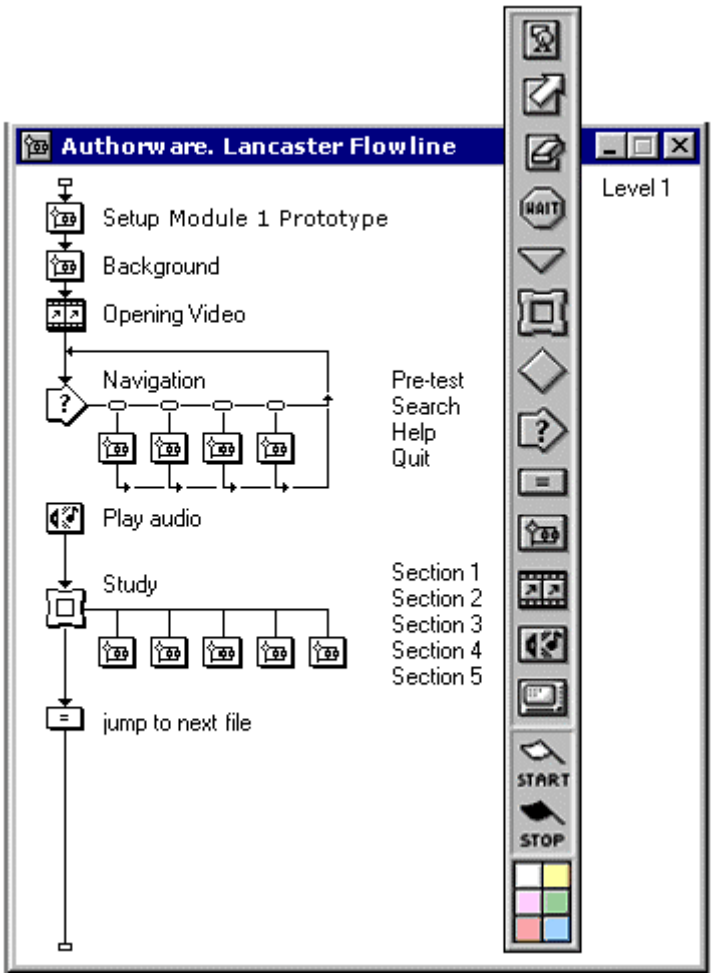


Figure 7. The Outline View in Macromedia's Authorware Program

Macromedia's Authorware program has been around since about 1992, and was designed with interactive training in mind. It incorporates an intuitive visual interface (that I believe to be modelled after AmigaVision, 1989). Dragging and dropping icons create a flowchart outline for the application being developed. The program automatically generates the scripting and programming language for various functions and events. You can even import a designed course created in *Designer's Edge* software.

There are a few features I particularly like about this program. The framework and navigation icons help to easily visualize and create rather complex navigation structures. This flowchart-like metaphor is ideal for educational design, as one can visualize a learner moving through the instruction, and easily modify or edit the program at any point. There are a number of pre-designed templates (sometimes referred to as *knowledge objects*) built-in, and there is the ability to create user-defined templates. These can help cut down on development time and automate the development process. Basic testing and scoring interactions are

included in these templates, which makes it a good choice for creating educational applications.

On the downside, there is a fairly steep learning curve involved, and the program assumes that the user knows how to create multimedia content. Authorware is an excellent choice for developing CD-ROM based applications, but can be rather sluggish when streamed over the Internet, especially if the content is media-rich. However, for 'drill and practice' applications, it performs well on a network, and the response times are quick. Authorware is compatible on both Mac and PC platforms.

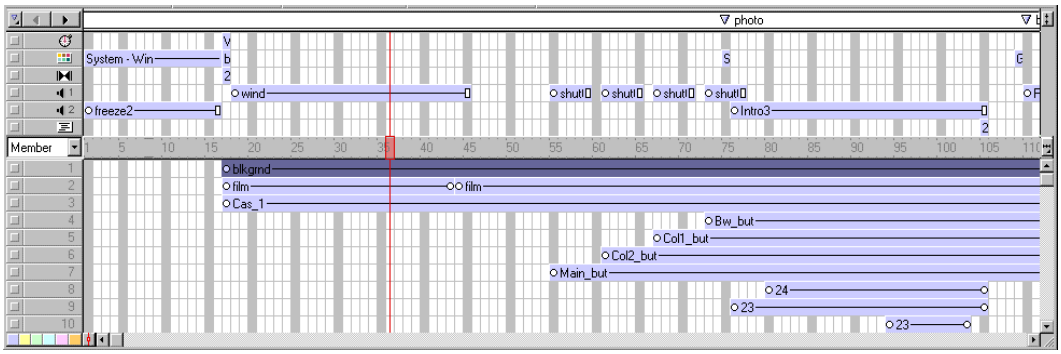


Figure 8. The Score Timeline Window from Macromedia's Director 7

Some authoring programs are based on a timeline metaphor. Such is the case with Macromedia's Director Program. I would estimate that about 75% of all commercially produced multimedia programs have been authored in Director.

Unlike Authorware, which is more of a multimedia assembly program, Director allows you to manage, edit, compress, modify or even create multimedia elements. The Studio version comes packaged with a number of integrated production tools, for recording and editing audio and creating and manipulating visuals.

The Director authoring metaphor is based on movie production, with a cast, stage and score. Media is imported into a cast window as a thumbnail, and then the cast members are placed onto a stage for positioning and interaction. The score (timeline) controls the sequence of events (Figure 8).

To create interactivity, there is a libraries menu with commonly used functions to control behaviours and assets. Behaviours include interactive elements such as navigation, user-interface widgets, controls, and timers. Beyond the libraries functions, Director also uses its own robust, object –oriented scripting language known as *Lingo*, which is very similar to Visual Basic and Java Script.

One of the things I dislike about this program is that it is sometimes difficult to place all of the many windows and elements on a CRT, especially when working on large projects that incorporate many media assets. The program does have a few management tools to help overcome this, but I have a 19-inch monitor and

still have a tough time with crowding (Figure 8). Using two monitors with a special video output card is one way to overcome this problem.

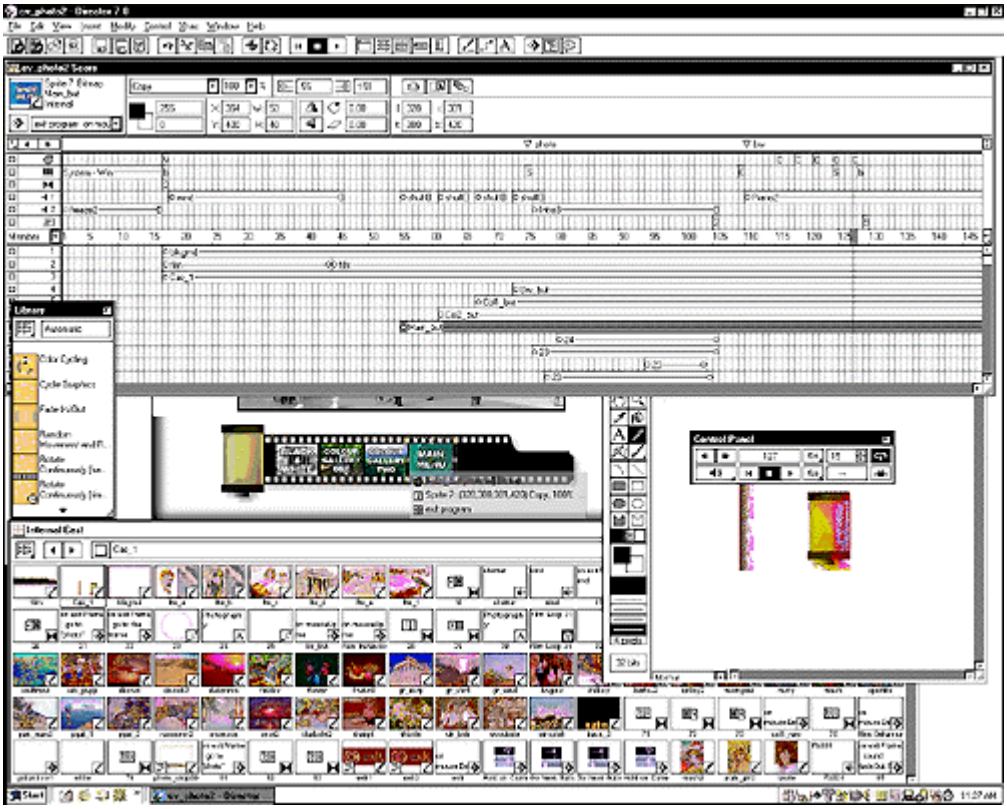


Figure 8. A Crowded Director Screen

As with Authorware, Director is an excellent choice for developing CD-ROM based applications, but can be very sluggish when streamed over the Internet as a Shockwave file with rich-media components. There is PC and MacIntosh compatibility, but the final movie or application has to be re-rendered for use on the platform other than the one it was initially created on.

The learning curve on Director is very high, and it would not be a practical choice for content experts. Even though it is not specific to education, it is still a proven industry standard, and a very versatile program with powerful features.

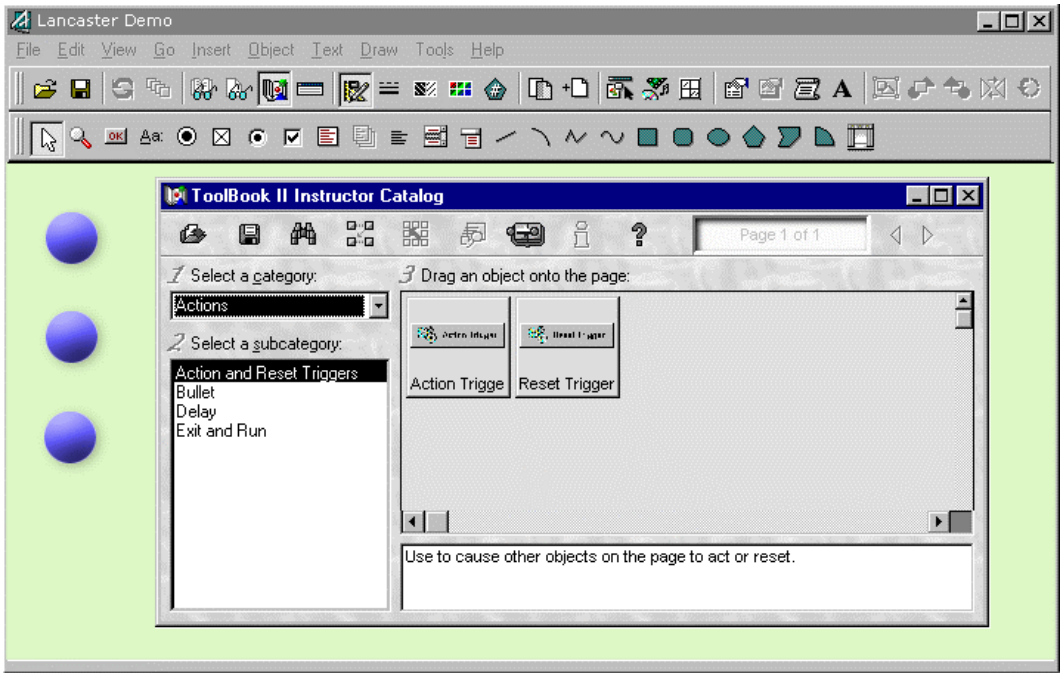


Figure 9. A Screen from Toolbook II Instructor

ToolBook II Instructor is a popular courseware specific authoring product used by professional developers, programmers and instructional designers. Its companion product ToolBook II Assistant is compatible, and aimed at educators who are non-programmers. People with various skill levels can use the two products simultaneously while working on the same project.

ToolBook utilizes templates, wizards, and pre-programmed catalogue objects, all similar to Director's libraries and Authorware's templates, which are accessible through pull-down menus and icons. It also has its own deep programming language, which can help create more customized and rather sophisticated courseware.

A ToolBook template is a pre-built area that can be customized by adding content to it. Catalogue objects extend and enhance these templates with built-in behaviours to handle such tasks as scoring a question, playing a media file, and navigating from page to page.

Personally, I find that ToolBook should be a lot easier to use than it is. Although it uses a simple book, page and object metaphor, I find it fails to deliver with a simplified and logical, easy-to-understand layout of its tools and functions. In some areas, I find it oversimplified to the point of causing much frustration in limiting design freedom, and at other times, extremely illogical and difficult to understand for the most basic design functions.

Like Director and Authorware, this is a product that was initially developed for CD-ROM applications, but also boasts a quasi-Internet/intranet/network delivery mode, which is once again impractical for networked media-rich content. I know from experience that you had better have a programmer close by if you are

developing any project with a substantial scope. There is no Mac compatibility. This is one of the few more powerful authoring tools specifically designed for educational purposes.

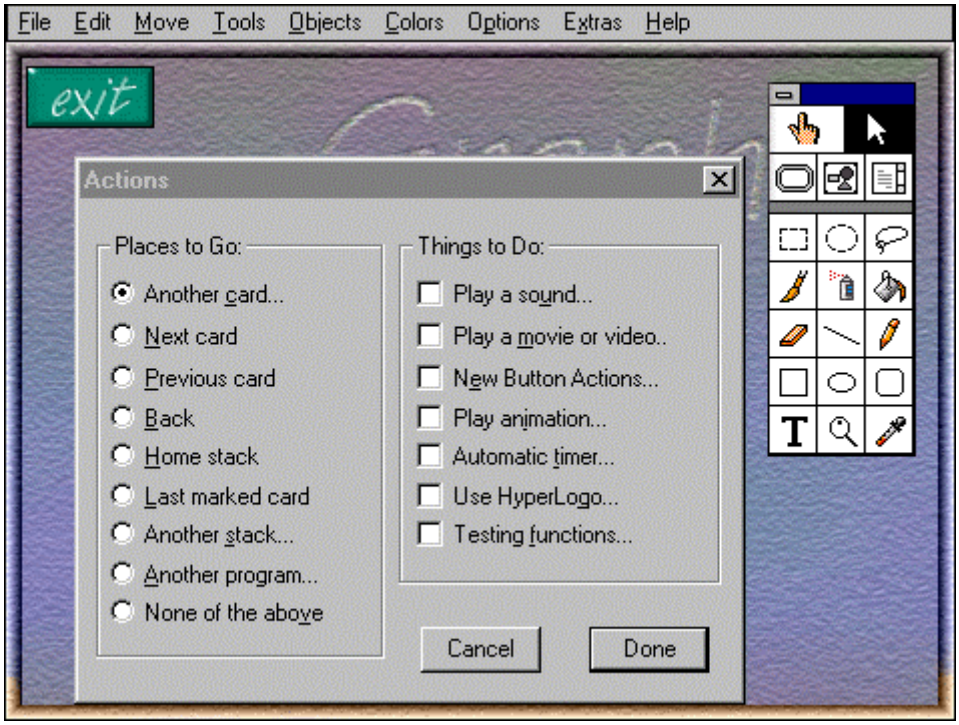


Figure 10. The Actions Menu Selection in Hyperstudio

Another authoring program that I feel deserves mention is Hyperstudio (Figure 10). In some respects, it is very similar to ToolBook, in that it uses menus and a card and stack metaphor that is very similar to the page and book metaphor. Unlike ToolBook though, its interface is extremely user friendly and logical. The author can develop the program by choosing either ‘what to do’ or ‘where to go’ functions from a menu (Figure 10). With this elegant simplicity and its low cost, it’s no wonder that Hyperstudio is used all over the world for student knowledge engineering. I have used it extensively for teaching multimedia literacy to students of all ages with excellent results.

For more advanced applications, Hyperstudio uses its own *Hyperlogo* scripting language. Unfortunately, testing, assessment and tracking ‘widgets’ are minimal in the program and need to be coded, and this may make it a poor choice for some content experts. Serious courseware development work may become tedious with the amount of coding required, although Version 4 does provide a few ready-made scripts. It may prove ideal for developing a quick prototype that illustrates basic navigation and presentational design. There is compatibility between Mac and PC platforms, but many of the media files (i.e. audio) do not transfer easily and may have to be reformatted or re-embedded. Hyperstudio is primarily used for CD-ROM based applications, but does have a browser plug-in that allows for online streaming.

I have reviewed a few popular authoring programs: Authorware, with its icon-driven, flowchart-like metaphor; Director, with its timeline and movie metaphor; ToolBook, with its menu-driven, page and book metaphor; and Hyperstudio, with its menu-driven, card and stack metaphor. All of these products include a royalty-free runtime module for distribution.

Selecting the proper tool has much to do with the delivery platform and the technology available at the learner-end. Any of these development tools would be good choices for CD-ROM and some networked applications, but due to the nature of the rich media content they typically utilize, and the bandwidth limitations of most networks and the Internet, they may fall-short, as they were not originally developed with networked learning in mind. This issue however, is constantly under development and improving. Ideally, the products produced with these types of authoring tools could be used in conjunction with online content, as they all support hyperlink and WWW compatibility. In other words, learners could navigate to or access specific files on the Internet or a network from URL's embedded in the base application on a CD. Accessing media-rich courseware on a CD-ROM to compliment online components would be another way to view more practical applications of the product these programs produce.

These tools can support learning management facilities such as student registration and tracking, and we can expect to see more integration of features that support the delivery of training over networks in the near future.

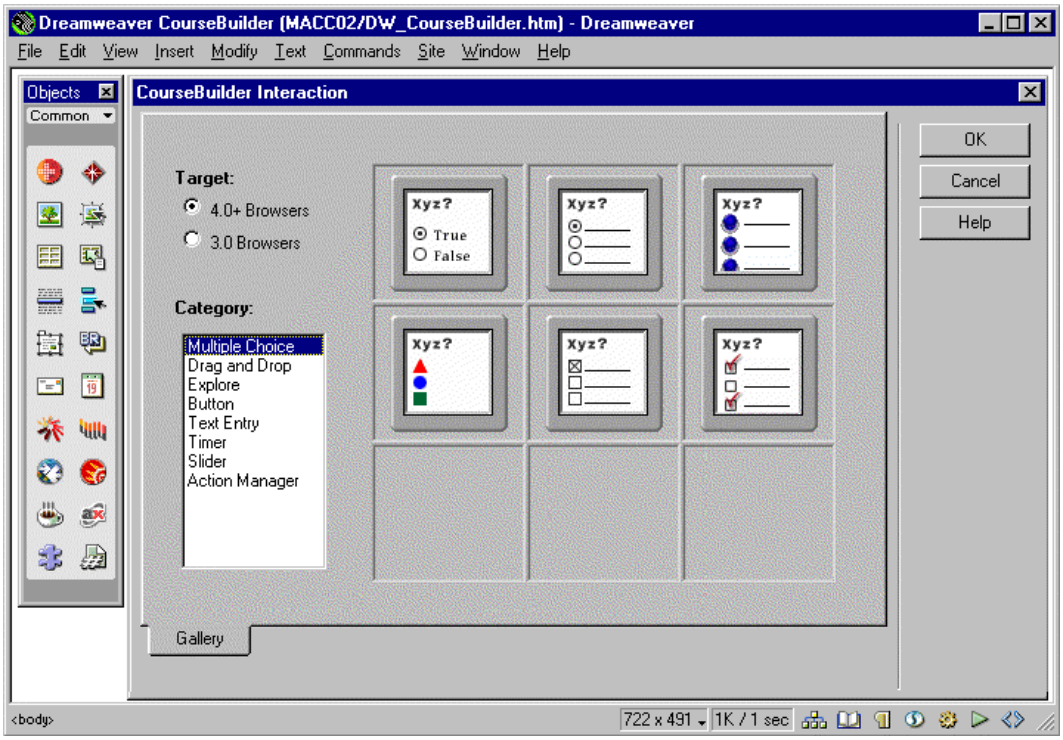


Figure 11. Macromedia's CourseBuilder Extension Module for Creating Learning Activities within an HTML Authoring Program

Other types of software to consider are those that provide learning extensions and functionality to existing web-based products. Macromedia has announced

recently that it has developed 'Learning Interaction Smart Clips' and other learning extensions to work with their core products. In other words, production software used to create web pages will now include 'widgets' for producing interactive learning activities (Figure 11).

Learning Management Systems

Now we consider a typical learning management tool or system that has been specifically designed for networked learning applications. WebCT (Web Course Tools) is one of the most popular tools available, which facilitates the creation of rather sophisticated World Wide Web-based educational environments. Educators at the University of British Columbia in Canada developed WebCT in 1995.

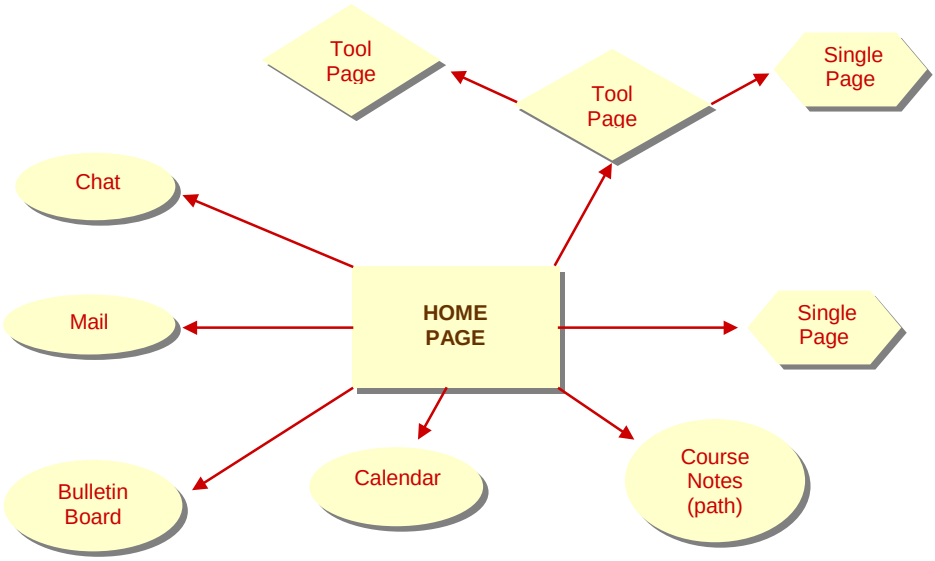


Figure 12. The WebCT Environment

Most learning management systems are based on hierarchies that branch out from a main page.

As the name suggests, WebCT provides a set of tools useful for a broad range of teaching methodologies that are consistent with new paradigms of teaching and learning. The way in which WebCT is designed makes it easy for course designers to create an organized structure for online instruction, and incorporate powerful features such as asynchronous chat, glossaries, a whiteboard, libraries and resources, course modules, testing, monitoring and tracking, and multimedia (Figure 12).

Other LMS packages offer essentially the same features, and include Blackboard, TopClass, ClassAct and Lotus Learning Space to name a few. What is critical is the way in which the content itself is designed. The most common error when these types of tools when placed in the hands of novices, is that vast amounts of existing textual information that already exists in curriculum is uploaded into the program, and dubbed an online course. It is important for an institution to

determine exactly how this type of tool is to be used consistent with its vision for teaching and learning, and provide the necessary faculty support for implementation. For content experts and teachers, it does provide an easy way to upload and present instructional content, and monitor student progress.

As a courseware engineering tool used by more experienced designers, the structure provided can be extremely labour saving, and provides consistency and standardization with a variety of courses in a large setting.

Figure 13. The Designer Mode in WebCT

WebCT can be accessed by: the course administrator who sets up and arranges blank courses; the designer(s) with full editing privileges (Figure 13); a grader or marker with limited grading and tracking access; or the student, with limited access.

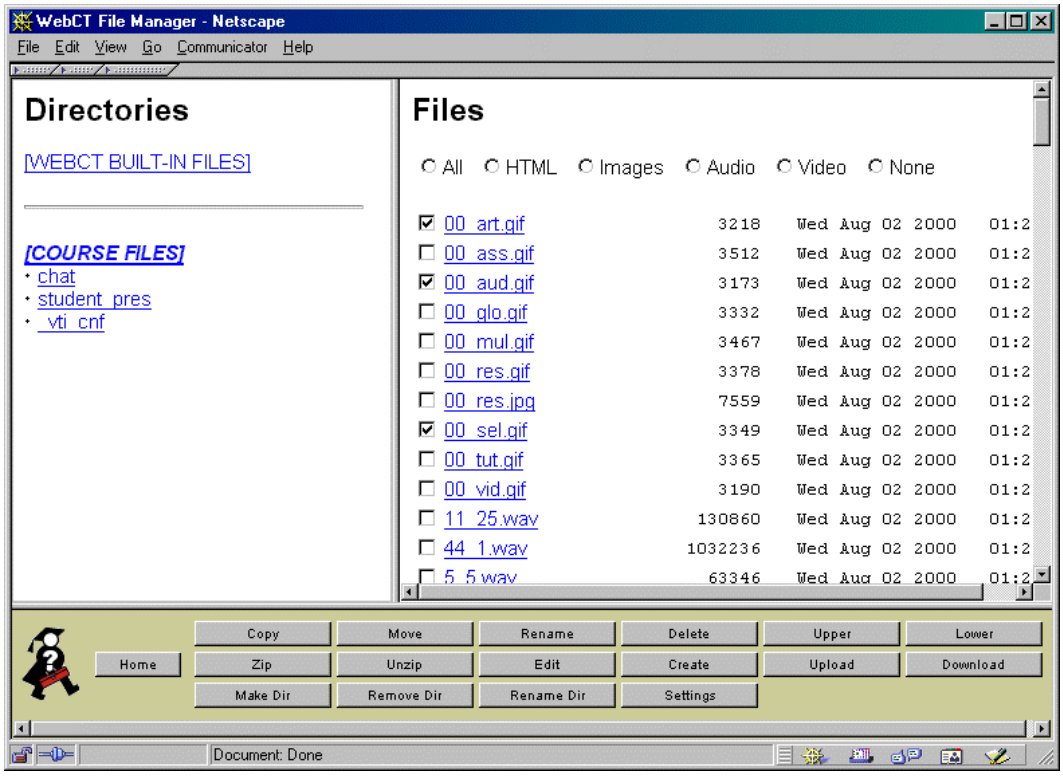


Figure 14. The WebCT File Manager Screen

There is the ability to edit or even create HTML code in pages, but for the most part, web pages are constructed outside of the environment and imported in through the file management facility (Figure 14). This takes some getting used to, and links to content and other components embedded in pages could become invalid if one is not careful with the file references during construction. The process of importing one page or multimedia element at a time can become quite monotonous. An easy solution is to ZIP all of the completed pages and resources in a hierarchy, and then import and UNZIP them in the WebCT file management section.

I was not impressed with the initial program and its clunky interface, which took some time to become familiar with. However, it was definitely the right type of tool at the right time, and it has undergone significant improvements since its inception. In fact, it has been so successful that a 70-million dollar equity investment in the company has recently been made, which will make it a world standard in education (2000).

The tracking facilities are lacking in that it cannot be determined how long a student was online and exactly what they did. My students caught on to this quickly, and learned that just clicking on an article reference might indicate to me that they had actually read it. The marking and student data resources are quite good, and give the instructor valuable information on student and class averages.

Another area of concern deals with the way global user and designer accounts are set-up with passwords. The interface and instructions are confusing, as most of the workshop participants I have worked with struggled with the process.

Setting up and maintaining identical courses that run at different campuses with different students can be a bit of a struggle, but there are a few ways to work around this.

Overall, WebCT may have had humble beginnings, but it is a very useful system that keeps on getting better due to its ubiquitous acceptance and use. There are a growing variety of classroom-ready content modules available from WebCT that designers and content experts can draw on for integration in their courseware. Major publishers have also started releasing content for use with WebCT.

We have looked at a number of viable tools that can help organize and automate the design, development and production of courseware. A problem with the use of these tools in educational settings is that they are pedagogically neutral, in that they quite often overlook the rudiments of instructional design and even presentational design. People may be too quick to accept fast and easy hardware and software solutions without realizing that there are bodies of knowledge to be considered that are specific to multimedia courseware engineering and education in general. Certainly, the quantity of multimedia materials produced has exponentially grown over the last few years. Improving and maintaining quality however, is the most important issue.

Many quality standards exist and are under development for educational multimedia and the computer management of learning. Some of these are being developed by:

- The Learning Technology Standards Committee of the Institute for Electrical and Electronic Engineers
- The Aviation Industry Computer Based Learning Committee
- The IMS Global Consortium Inc.
- The European Community Information Society Standardization System for Learning Technology

Products from a variety of suppliers that meet international standards will allow users to deliver and record usage of training and education (Guidelines for Managers, 2000).

PLANNING AND PROJECT MANAGEMENT IN MCE

DEFINING QUALITY

In part one of this paper, I have established that beyond technical levels, defining multimedia courseware quality in measurable terms is not easy. I briefly discussed what types of attributes a good finished product should have in order to establish a working specification that it understood by all parties. Comparing this type of end-product quality standard to the organizational context in which it is created

and used can create a basis for establishing and perhaps measuring what quality is. However, it usually comes down to what the stakeholders consider quality to be. A project manager must set levels of quality that are consistent with what is appropriate and achievable for a given purpose to reach defined results (England & Finney, 1999). In many instances, the stakeholders will have to be shown or told what is attainable, and this is where an initial prototype and rapid prototyping can prove invaluable. Quality output may not always be compatible with fast and economical prototyping, but an agreement with the stakeholders on what the content will be and how it will be presented will enable a project manager to regulate the various control mechanisms in the process.

Aside from product and process quality issues, there are technical considerations. These could include the ability of the courseware to run on multiple platforms. Even with the ubiquity of PC platforms, there are many MacIntosh users out there. In fact, most production houses I know of produce the majority of content on a Mac, and then bring the components over to the PC for assembly. Multi-platform products can cost more to produce, especially if authoring tools are not cross-platform compatible. Technical communication standards and interface standards are also considerations in ensuring technical quality.

The typical development time to delivery ratio metric in the industry has many faults, and does not prove very reliable in measuring productivity or estimating results (Marshall et. al, 1995).

TIME + BUDGET + RESOURCES = QUALITY

There are many variations to the above formula. Project management typically considers the issues of cost, schedule and quality. For example, a product that is needed in four weeks with an unlimited budget may be just as successful as one that had a very limited budget but an unlimited time frame, or one with a limited budget and timeframe but virtually unlimited resources. If any one of the elements in the formula is decreased, one or both of the others usually need to be increased. Establishing exactly what the balance is can help to establish the parameters for courseware engineering project management.

We have seen that that a MCE process goes through a series of phases and steps. Within each of these, a number of people issues can arise that affect the relationships among key players and the final outcome. The variety of backgrounds and skill sets of the people and managers involved can vary approaches so widely that conformity can never be established, and generic formulae or metrics never substantiated.

Project management is quite different from operational management, in that projects are typically temporary undertakings that generate a specific product or outcome. Tools and techniques represent the hard-side of management, while the much harder to quantify and control people side represents the soft-side of management. Good planning, teamwork and leadership are essential to ensure that a project is completed on time and on budget to the required quality. The project

manager's responsibility is to coordinate and integrate all of the hard and soft resources to meet these criteria.

PLANNING AND MANAGING THE PROCESS

Uncertainty and change in a project represent significant challenges. Even with careful planning, the more innovative and unique a project is, the greater the chances of unforeseen change. The whole notion of a MCE process is to provide some semblance of structure in which to operate and solve problems, and these structures are rarely absolute, therefore, one should anticipate changes that were not part of the original plan.

The organizational context in which a project is being developed can be a significant determining factor affecting management strategies. A non-commercial or educational environment may develop courseware in a leisurely pace with few time constraints. There may even be more tolerance to design flaws as well, although it should be noted that students today expect robust software. A commercial developer however, will be focussed on minimizing development time and maximizing profits will little room for error or reworks (Marshall et al., 1998). When used in a commercial environment, the milestone approach is much more rigid and unforgiving. Imposed deadlines can be good motivators, but also good sources of creating stress. Producing the same type of product within a similar environment will help establish a system that works for a particular faction.

By taking the courseware product quality, organizational context and the development lifecycle into consideration, a project manager can begin estimations on the amount of time it will take and on the resources required to complete each cycle or activity. There are a number of useful tools available to assist in the planning and tracking of project activities.

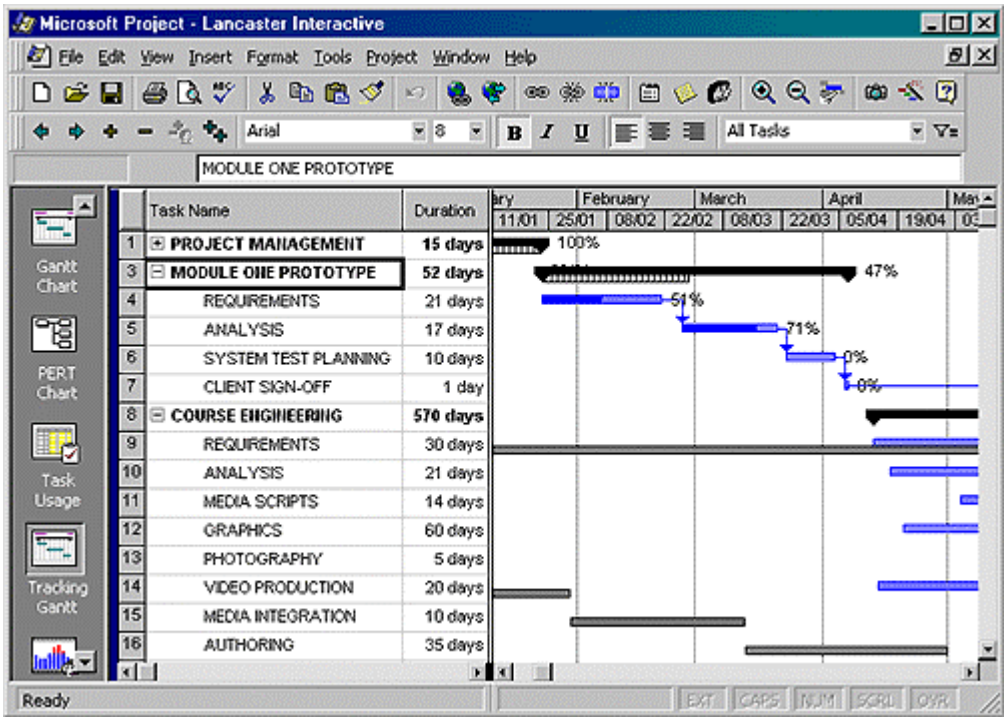


Figure 15. The Tracking Gantt Chart in Microsoft Project Manager

Microsoft Project Manager provides a number of useful features for organizing, planning, and tracking project development, most of which are based on time dimensions. A Gantt chart (Figure 15) graphically displays bars representing the time relationships of the steps in a project. It illustrates the flow of activities in sequence including those that are in parallel. Once initially completed, a Gantt chart conveys the estimated minimum total time required for project completion and all of the steps involved.

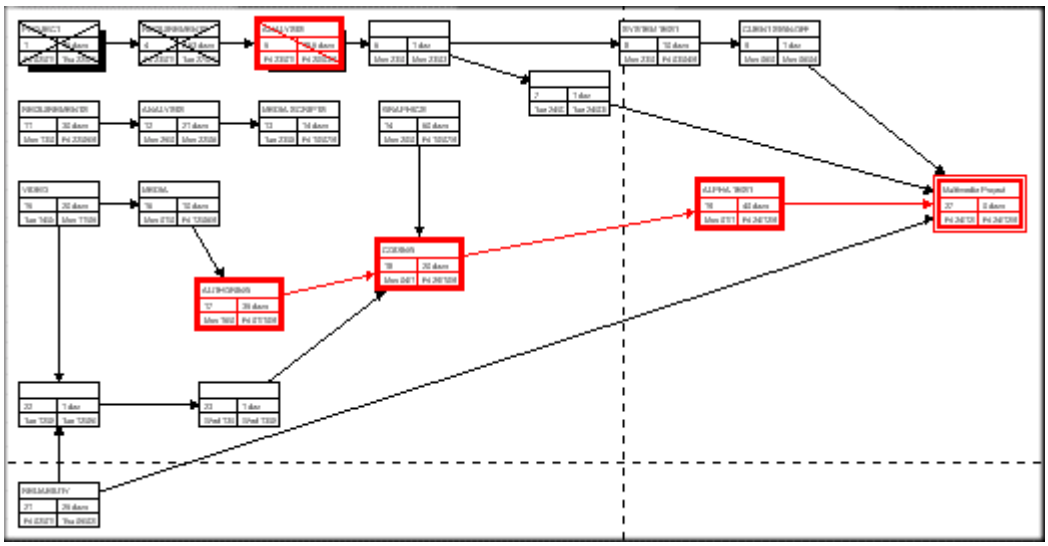


Figure 16. A PERT Chart Illustration from Microsoft Project Manager

The relationship between time, cost and quality are not linear. When there is great interdependency among the various activities involved, the same information on the activities can be displayed much more effectively in the form of a PERT (Program Evaluation and Review Technique) chart. The relationships between events, activities and non-activities (no work required but still an essential step) are illustrated along with time estimations (Figure 16). This type of network or critical path analysis is much more specific to MCE than a Gantt chart, which is usually associated with project management in general. Another program that uses a more ideal type of critical path graphical representation is Allen Communication's *Manager's Edge* program, which also interfaces with their *Designer's Edge* Software.

A project manager needs to have enough organizational context experience to know how to accurately estimate the time and costs involved for each phase and step of a project. With viable estimates, monitoring the actual progress will be easier and the project can be kept on schedule and on budget.

BUILDING AND MANAGING A TEAM

On the soft side, I feel that one of the reasons why multimedia projects fail so often is because of the unrealistic mismatching of skills and roles of the actual development and production team. Various components such as audio and video may also have their own design strategies. For example, producing interactive video components may require expertise in many areas including scriptwriting, directing, lighting, camera work, editing, animation/graphics and drama or acting. I have worked on many projects that incorporate design processes within design processes to accommodate the various media involved. For illustration purposes, the following is a rather utopian list of the positions and expertise that might be required in a large project of significant scope.

Accountant - keeps track of budget, costs.

Actor/talent - demonstrates or presents content; performs

Assessment teams - to collect, analyze and measure qualitative and quantitative data.

Audio Technician - records, edits and mixes sound; digital conversions

Director- supervises and directs actual production sequences including rehearsals, recording, programming, etc.

Content expert(s) - provides the instructional content; preferably an experienced tutor or the combination of a tutor and a subject specialist; experience in curriculum development and alternative delivery methods a bonus; students may also be called upon to help determine content input

Graphic Artist/Designer - produces most aspects of presentational design: drawings, backgrounds, objects, interfaces, etc.

Instructional designer - develops and/or interprets stakeholders' and learner's needs and requirements; applies pedagogy to innovative electronic learning environments to ensure instructional outcomes are met

Researcher/historian - does content research and/or acts as journalist; maintains copies of all project documentation

Media/New Media Specialist - integrates video, audio, graphics and/or other various media into the program; applies digital methodology to analogue components

Marketing team - to promote and/or market the product

Musician- writes, plays, chooses and/or produces music to be used

Narrator- does voice-overs of content

Photographer - takes still photos; retouching and digital conversion

Producer/Associate Producer- oversees the production process, procures releases and handles copyright issues.

Production Assistant - assists with the Producer's activities and scopes out and coordinates locations for recording, etc.; develops production schedule, rents equipment

Product Beta Tester/Editor - looks for programming bugs, continuity problems, etc.

Project manager - coordinates and oversees all aspects of production and development; educational experience, technical multimedia and new media production skills, and HR and project management experience a must

Programmer/IT Specialist - programs or codes for interactive computer environment; provides the technical expertise necessary for GUI development; may work in collaboration with a multimedia author; data management; may provide instructor and/or learner support and training

Proof reader/editor - checks for spelling, grammar, and clarity in all written and/or audio/visual material

Writer- writes all content, scripts, presentations, and accompanying documentation; works in conjunction with content experts in interpreting form

Obviously, a team of this magnitude would be far from reality, and out of reach for most educational institutions. It would be practical to assume that team members may take on more than one role for a project, and that outside consultants and expertise may be used when necessary and/or affordable. A stripped-down but functional version of the list of team members may read something like this:

- Project Manager
- Content Expert
- New Media Specialist
- Multimedia Programmer/Author

Whatever the size of the team, all of people involved may have different but relevant perspectives on how the content should be structured and treated (England & Finney, 1999). A lack of situational awareness among team members can lead to severe consequences, and clear communication on roles and expectations is essential to avoid disputes on how particular tasks should be accomplished. I have personally found (painfully) that giving a respectable amount of creative freedom and responsibility within defined parameters or even areas of expertise can keep the team on track. Simply asking participants what they would prefer to do and what their perspectives are on their role is an excellent place to begin to establish and maintain a role assignment list for the personnel and respective activities required.

To be effective in adaptability and coordination of action, individuals will also need to identify and share knowledge about what other team member's functions and role responsibilities are, and must know when and how to integrate with them. A more defined coordination strategy to deal with the task(s) to be accomplished can then be developed with specific roles and responsibilities for each participant. Shared strategic mental models will enable participants to understand where they fit in and how they can contribute accordingly. Brainstorming sessions and the contextual assigning of roles would be a good way of beginning this process.

A cooperative environment should include mechanisms to encourage perpetual communication and feedback, which is necessary in order for individuals and the team to deal with problems that relate to procedures, actions or knowledge in a timely manner. A cooperative environment should be designed that allows a development team to discuss and realize what the overall goals and objectives are, and what tools and knowledge are available. Determining the types of declarative and procedural knowledge that are present or required within a group can help set parameters within the task domain. Intervention may be required to set up training strategies such as cross training, task practice or simulation to increase the awareness or skill levels within the group. However, most developers will learn the subject content by working with someone who knows it, just as a subject matter expert will also learn about instructional design. This osmotic effect creates excellent opportunities to improve situational awareness in a courseware development team. It also promotes quicker movement up the corporate maturity framework.

Task Name	Work	Details	Week 5 S	M	T	W
PROJECT MANAGEMENT	**360 hrs**	Work				
MODULE ONE PROTOTYPE	**573 hrs**	Work		15h	5h	8h
REQUIREMENTS	41 hrs	Work		15h	5h	
Laurie	*15 hrs*	Work		7h		
Allan	*21 hrs*	Work		8h	5h	
Hank	*5 hrs*	Work			5h	
ANALYSIS	284 hrs	Work		0h	0h	8h
John	*12 hrs*	Work		0h	0h	8h
Bill	*136 hrs*	Work		0h	0h	0h
Hank	*136 hrs*	Work		0h	0h	0h

Figure 16. Scheduling Tasks and Assignment Roles in Microsoft Project Manager

Software tools can aid with assigning individual and group roles and monitoring progress (Figure 16). A good project manager will not arbitrarily assign roles, and will encourage feedback and communication between all participants

Considering the amount of expertise, coordination and production that could be required, it's no wonder that many multimedia projects go over budget. Artistic and technical trial and error seems to be a big factor in academia, where these resources are limited, thus creating time factors that are more forgiving. Rather than discover that certain skills are required in the middle of a process, the project manager must scope out exactly what is required and expected, and how it can be accomplished.

Managing and controlling a MCE process requires special attention to any factor that may hamper the time, cost and/or quality of the product. These factors include variances with the client, work environment, market forces and the development team.

CONCLUSIONS

I have examined a number of facilitatory resources, including typical practices, methods and tools that are required to provide an infrastructure for developing a multimedia courseware engineering approach. We can borrow from the established domain of software engineering, as the production, testing and maintenance stages are similar in courseware engineering. Beyond software engineering frameworks, well-orchestrated analysis and design strategies and a well-defined process to formulate these strategies into a quality product are a critical part of developing a viable MCE framework.

By attempting to define generic issues relating to CME products and related processes, we can begin to establish what a quality courseware product and process is considered to be. On a macro-level, knowledge-rich tools can assist in the development of instructional materials, but they are typically of limited function and use, and therefore categorized as weak automation tools. On a micro-

level, one can appreciate and perhaps apply some of the pedagogical strategies and recurrent practices that are associated with formulating and using these types of tools.

MCE is a complex environment, and developmental models can provide developers with different levels of abstraction in a framework in which to solve problems. There are many models available for developing courseware, which are basically variations on analysis, design, production, evaluation, revision and management activities. Analysis and design in MCE are critical and deserve serious attention in identifying the appropriate learning methodologies as well as analysis of the subject matter and/or skills to be learned. This should be consistent with how the instruction fits into the overall system.

Initial requirements should be specified clearly so that analysis-design phases are not part of an endless cycle. Initial and rapid prototyping can increase development team situational awareness and give stakeholders a better understanding of what the product will be. Various software tools can organize and automate some of the courseware design process. The way in which a development process is perceived and applied may be quite unique to its organizational context. Various methodologies can become structured and recurrent through maturity in a particular working environment. Basic development models can allow a faction to develop more sophisticated, customized models that better reflect the reality of a specific work environment.

Authoring systems enable the production of interactive multimedia courses. Learning management systems support the management of learning in an organization. Both types of tools can be invaluable in constructing protypes and the courseware itself, and can help organize and automate design, development and production. I looked at a variety of these tools to gain an understanding of their operational principles. The wide proliferation and simplification of these tools has enabled more people to create multimedia-based instruction. However, these tools often overlook the rudiments of instructional and presentational design, and instruction may be required as to their ubiquitous use.

A project manager must set levels of soft and hard quality that are consistent with what is appropriate and achievable in a given context for desired results. Project management typically considers the issues of cost, schedule and quality, which are difficult to qualify in MCE project management. Good planning, teamwork and leadership and an awareness of organizational context help ensure that a project is completed on time and on budget to the required quality. The use of management software that produces critical path analyses and enables scheduling and monitoring can be an invaluable tool for coordinating projects of substantial scope. Finally, a cooperative development environment should include mechanisms to stimulate situational awareness in order for individuals and the team to deal with problems that relate to procedures, actions or knowledge.

BIBLIOGRAPHY

Bell, B (1998) Investigate and Decide Learning Environments. *Journal of the Learning Sciences*, 7(1), 65-105.

Bell, B (1999) Supporting Educational Software Design with Knowledge Rich Tools. *IJAIED*, 10.

Boehm, B. (1988) A Spiral Model of Software Development and Enhancement. *IEEE Computer*, 10(5), 61-72.

Bostock, S. & Drummond, P. (1996, revised 1998) Courseware Engineering -an overview of the courseware development process, Keele University, Staffordshire, U.K.

de Diana and van Schaik (1993) Courseware Engineering Outlined: an Overview of some research issues. *Educational and Training Technology International*, 30(3), 191–211.

England, E. & Finney, A. (1999) The Background: multimedia and project management. Chapter 2 of *Managing Multimedia: Project Management for Interactive Media*. Addison Wesley.

England, E. & Finney, A. (1999) Rights, copyright and other intellectual property. Chapter 17 of *Managing Multimedia: Project Management for Interactive Media*. Addison Wesley.

Goodyear, P. (1993) Foundations for Courseware Engineering. In R. Tennyson (Ed) *Automating Instructional Design, Development and Delivery*. Berlin: Springer Verlag.

Goodyear, P. (1994) Infrastructure for Courseware Engineering. In R. Tennyson & A. Barron (Eds.) *Automating Instructional Design: Computer Based Developments and Delivery Tools*. Berlin: Springer Verlag.

Guides for Managers, Practitioners & Researchers. Technology Based Training & On-line Learning (2000): An overview of authoring systems & learning management systems available in the UK. A Lifelong Learning and Technologies Division Publication. DfEE.

Hsieh, P. Y., Halff, H. M., & Redfield, C. L. (1999) Four Easy Pieces: Development Systems for Knowledge Based Generative Instruction. *International Journal of Artificial Intelligence in Education*, vol. 10, 1 - 45.

Marshall, I.M., Samson, W.B., Dugard, P.I. & Lund, G.R. (1995) The Mythical Courseware Development to Delivery Time Ratio. *Computers & Education*, vol. 25, no. 3, 113 – 122.

Multimedia: Making it Work, Second Edition (1994) Tay Vaughan, Osbourne McGraw-Hill, U.S.A., ISBN 0-07-882035-9

Reigluth, C.M. (1999) *Instructional-Design Theories and Models*, Volume II. Hillsdale, NJ: Lawrence Erlbaum, ISBN 0-8058-2859-1

Spector JM, Gagné RM, Muraida DJ & Dimitroff WA 1992. Intelligent frameworks for instructional design. Educational Tech. October 21-27.

Survey of Perceptions and Attitudes towards Learnware Quality in Canada (1998), Office of Learning Technology and Human Resources Canada & Knowledge Connection Corporation