ENGINEERING A TECHNIQUE FOR ONLINE COURSES DEVELOPMENT

Dr. Akram Salah

Department of Computer Science
North Dakota State University
IACC 258
Fargo, ND, 58105-5164
Email: akram.salah@ndsu.nodak.edu

Abstract

Designing a course for online offering is different from classical teaching. This paper uses a proxy benchmarking by borrowing techniques from software engineering that are used to develop software. We argue that both software and course development are intellectual in process and product. We define an evolutionary development lifecycle methodology based on the tasks of analysis, engineering, content development, and feedback. Within engineering, we define structural design, architecture design styles, and navigation design techniques. Then we define some management challenges for a course as it is offered and maintained.

The paper sets some framework for course design making use of modular approach. This allows for making use of a hybrid structure of material to allow for the instructor, or the course developer, to mix and match many components in an easy to follow and easy to maintain fashion. It is a step in developing a platform for online course development.

1. Introduction

The movement from the classical course offering, i.e., classroom with a direct contact between the instructor and the student to the online offering, either as a complete online or the partial online as a support to the course, created many differences in the course design and material structure. This move is very much similar to the move from sequential access information processing to direct access information processing. In the classical course offering, the material is organized by the instructor in an ordered form, with lectures, units, exams, assignments, etc. The plan is predefined by the instructor and is carefully followed by the students. The move to use black boarding to post material designed by the instructor still carries the same philosophy. The plan is set by the instructor and followed by students in a synchronous mode.

The real value of using the digital media, with computers for storage and networks for communication, for course offering is to make advantage of characteristics of computer and networks of direct access and asynchronous mode of handling the material. This would allow a student to access the course material in his/her own pace. Each student may have an order in viewing the material that is different from other students. This is preferable pedagogically, however, it is not possible in the synchronous method. However, it could be achieved in the digital media given that the material is organized to do so.

Another feature that can be facilitated through the digital representation is the direct referencing to support material. A student may want to refer to the glossary or to the index of material to access related material for reference. The glossary and the index are types of references that are related to the whole course, however, there would be other reference possibilities to related material with the course as a student is viewing or answering the material in one part. A reference could be to information within the same course or supporting material in other courses.

We can assume that problem is coming from interpreting the material in a book dependent fashion, or may be in a class related fashion. Classes and books are organized in a serial arrangement. We need to reengineer the process of viewing the material and with that we need to have a design methodology different from what we are currently doing.

In this paper we are trying to define a course development methodology, course engineering, to provide a framework, for systematically develop material for a course to be offered online. The methodology is borrowed from software engineering in developing software products and web applications. We believe that both course material and software are sharing the characteristics of being intellectual in nature, both, the product is an intellectual and the process of development is intellectual. We also believe that the course offering now is more like first generation of programming where programs were presented as a long sequence of code. The second generation was based on modular and structured concepts. Thus, we take the position that the second generation of online courses would be applying more modularity and structural characteristics.

The structure of the paper is as follows. In section 2, we overview some software engineering concepts that would be useful as course characteristics. In section 3, we discuss structural design. In section 4, we discuss architecture design and architecture styles. Then, in section 5, we suggest a methodology for course engineering. In section 6, we discuss some of additional management issues.

2. Software Design Concepts

Since our approach is based on software engineering, we start by viewing some of software engineering concepts that are believed to be of interest to course development.

Structure

In dealing with complexity of the problem or the solution, we tend to decompose the problem into sub-problems and further to small components with a clear definition of the relationships among components[2]. We refer to the components and their relationships as the structure, of either the problem or the solution, i.e., the software. In courses it is the same concept. The knowledge content presented in a course is a complex body and to present it we tend to decompose the content into modules and then further to components. To keep the structure of the course we represent the relationships between components in a standard way.

It is noteworthy to say that in course development we have more than one type of structural characteristics and thus more relationships. On one type of relationship, we may explain the same topic on more than one level of depth. On another type, we may have more than one aspect to explain for the same topic, e.g., theory, applied, techniques, etc. On a third type, we may have more than one approach to explain the same topic, such as using examples, case studies, plain explanation. Moreover, we have the content itself, illustrations, and tests and evaluation. So the nature of the courses reflects structure and complexity on more than one dimension.

Reusability

Reusability [4] is the extent to which a program, or part of a program, can be reused in other related application to the packaging and scope of the functions that the program performs. In course development, and education in general, reusability is very much dependent on the overlap of content between courses and course components. Every course depends on components from other courses and within the same course; each component has dependence on each other. This dependence usually reflected as components that are referenced from more than one component.

It is important to relate the concept of reusability with the level of granularity of components. The less the size of a component, the easier and more frequent one can reuse a component in various courses, or refer to it from various parts of a course. It should also be noted that the usability of component, or reusability, could be within one course or across courses.

Modularity

A module is software is a named logical component that is potentially usable in more than one part of the software, or potentially referenced in more than one situation within an application. In software modularity [2,4] depends very much on functional dependence of a program component. In course design, a module is part of a course that has a clear objective. However, we will refer to a module as an independent component that is presented separately or can be referenced separately.

Again, it is noteworthy to connect the modularity with reusability and structural nature of the whole course, all the three concepts are related. The ability to break the course content to components with clear relationships would contribute to its modularity and definitely to its reusability. However, modularity in courses is part of the process of explanation. In fact, it is more natural in courses than software.

Maintainability

Maintainability is referring to the ease, in terms of effort or time exerted, of which an application program can be maintained after delivery. However, maintenance involves more than one task; usually include fixing errors, enhancement and adaptation. Extendibility, which is related to the maintenance, is defined as the degree to which architecture, data, or procedural design can be extended. We assume in course development that maintainability is an important quality. A course is taught many times and every time it is taught, we try to improve the content or the method of presentation. We should not reconstruct or redevelop the course material every time the course is offered. In fact this is one of the main advantages of using digital media to store and present course material. This would make the effort in development be high in the first time then the effort would be mainly on improving the course material every time it is offered based on the feedback from the students and the performance of the class.

Thus maintainability in course design is the ease of which we can expand or change course contents, or structure, after it is developed the first time without having to redesign the whole course.

Encapsulation

Encapsulation is a feature in object programming. It refers to the ability to encapsulate data and operations that work on the data in a single package. We may refer to the knowledge characteristics and doing characteristics of a class, i.e., what the class knows and what the class can do. In courses we can think of a module of a course as an object. The knowledge content would be what the module knows while the testing or assessment is what the module can do. That is, we encapsulate the tests of each module with the knowledge. This makes it easier to change one module as one unit, if it is needed.

3. Structural Design

The purpose of structure design is to decide on components of a system and their interrelationships. It is of special importance in complex systems since to deal with complexity, particularly for development purposes, we have to break down the system to components, that is decomposition process. A component is certainly simpler than the complete system. Associated with structure design is the notation such as Structure Diagrams, UML structure chart, HIPO, etc.

3.1 Course Structure:

In designing courses, the first step is to decide on the various modules that will comprise the course. Each module is further decomposed to smaller units. However, usually, associated with that is the time frame assigned for each module and unit with a review on each of them. Then a decision on refinement for the components, and thus the whole course structure, is concluded.

In this part of the design, we are not changing what is already done in course design; however, we make an analogy between this concept and object modeling in software development. If we can view each module in the course as an object, then we can think of the course layout as an object model. The whole course is a set of modules, thus we have one class. The attributes of the module are its name, length in number of class hours, level, process (practical, theory, or abstract), (lecture or lab), etc. A module is an object, i.e., it is an instance of the class module. Relationships among objects are class-subclass, aggregation, or related.

We can use some of the documentation/design tools of object orientation, namely CRC and Object diagrams [3]. CRC cards (Class-Responsibility-Collaboration) are used to document the responsibilities and collaboration of classes. In our case we use them to document the responsibilities and collaboration among modules within a course. The responsibility of a module can be both *knowing* and *doing* responsibilities. Knowing responsibility describes the information content of a module. Thus, it describes the purpose of the module and the method it provides the knowledge, i.e., its role in the learning process. The doing responsibility is the testing and assessment of the factors of completion of a module, thus the prerequisite of advancing to another module. Each module has a card where it contains its description, the purpose, and other modules that are related to it.

The Object Diagram is a diagram that documents the static structure of objects. It contains objects and its relationships. Of special interest is the aggregation relationship, which represents some modules being part another module. The second special relationships is the related to or the collaboration. This relationship represents the connection of the content of two modules in a way that makes the connected information contribute to each other.

3.2 Module Structure

A module is further decomposed to components. Each component has a purpose and content. The component is represented as a flow of knowledge stream to explain the topic. Within this part the use of illustrations, examples, figures, or diagrams to explain the concepts are used. Thus, to describe this part we can use the object concepts once more. Another important component of the module is the evaluation or assessment criteria of the module. This should result in constructing tests for the student to pass this module and advance to another.

To document this part of the structure we use again object diagrams. However, in this particular structure we distinguish the boxes in the structure diagram based on the type of information within the component. Thus, we have components being text, diagram, illustration, animation or video, questions, problems, case, etc.

4. Architecture Styles

We assume that each module is going to be presented as a set of components each component explains one particular aspect within the module. Each component forms a knowledge unit. Each knowledge unit explains one aspect using a process of explanation, i.e., either theory, applied, method, etc. The aggregate knowledge comprises the whole knowledge body of the course.

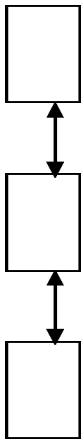
The arrangement of those components within the whole module forms the architecture of the module. Thus, the architecture is the way a module is built from its components. It may be not obvious to students when they are studying, but it reflects a plan of material. The decision on architecture is part of the instructor plan for explaining the material. However, it is also part of the student's studying style that he/she would like to follow a certain order. The challenge for the instructor is to present the material in a way that meets as many students' styles as possible.

One of the main advantages of online course offering, or support, is to allow various students to cover course material using their own style. Thus the choice of the architecture of the module is the key characteristic that allows, on one hand, the explanation process of the instructor and, on the other hand, allows the flexibility that provides for the student to navigate the material in various ways.

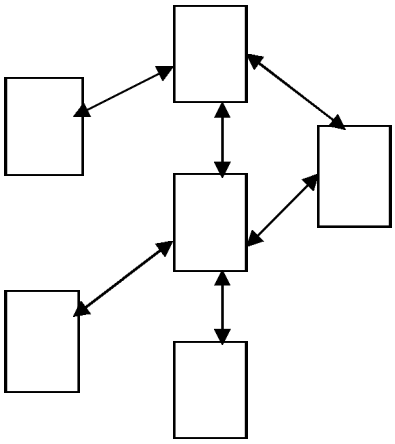
We identify here four architecture styles, linear, hierarchical, parallel, and network. Those four styles are recognized from the web site designs. We also suggest that part of the design process is for an instructor to choose a style for the architecture of presenting the material. This would make it more systematic to develop the online educational material. In this section we overview the four styles, then we comment on the appropriateness of the four styles.

4.1 Linear

In linear architecture the components of the module are presented in a sequential fashion. Thus concepts are arranged in a sequence, each one presented as a component. The sequence of the components follows the logical flow of knowledge within the module. A student starts at the top component and progresses to the next. We can envision that each component has “next and back” controls. An instructor may have a conditional advancement to the next component, e.g., through answering questions or passing a test. However, it is part of the design and development of the course and it does not affect the architecture style.



Linear Architecture

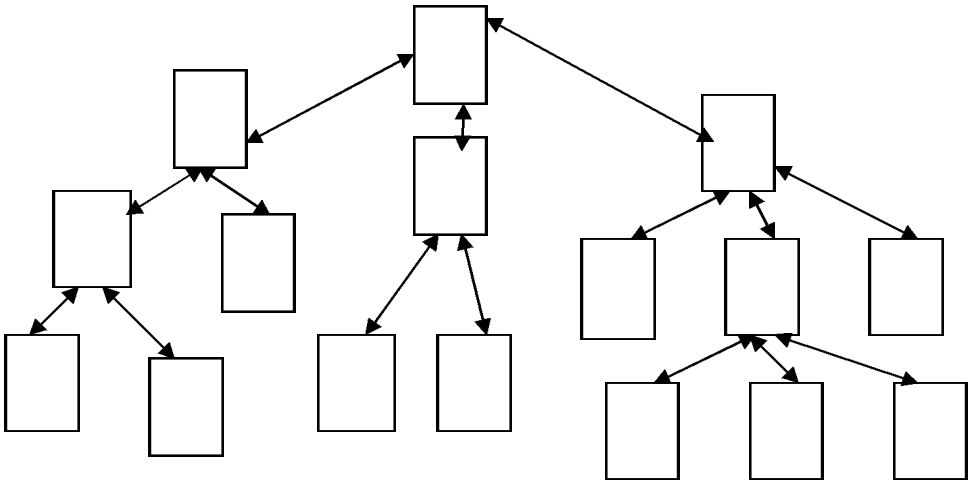


Linear with Diversion

A variation from the linear architecture is linear with deviation. In this style a component may include supporting material such as a diagram, illustration, or an example. In this case, the supporting material is connected to the component such that the content is linked to the supporting material in a way that would make it sub-links to the main flow of thoughts. Thus, viewers of the material get to the illustration from the component and then back to the same component.

4.1 Hierarchical

Hierarchical architecture can be also called tree. In this architecture style components of the module are arranged in a tree-like relationship. The root of the tree is the starting component. At the end of the starting component a viewer has the choice to advance to one of a set of optional topics to proceed. That is there is more than one way to understand the material, i.e., there is more than one logical line of thinking. However, each component has one parent component. This, the back link for any component is unique.

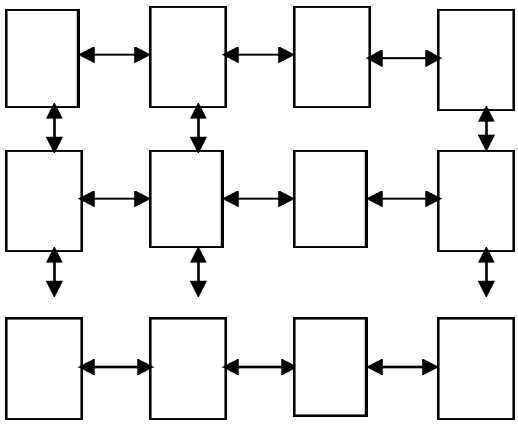


Hierarchical (Tree) Architecture

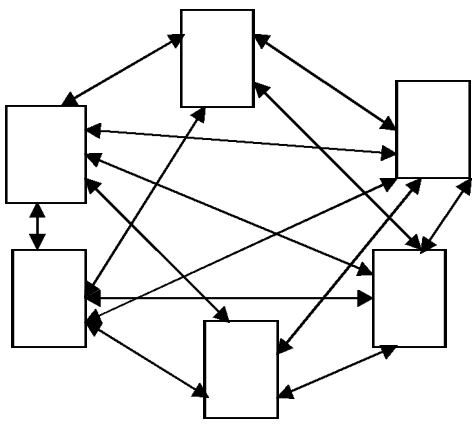
The hierarchical style gives a viewer more than one way of covering the material. There is more than one option for a viewer. He/she can go in the depth of the tree, in fact each path from the root to a leaf, defines a line of thinking. However, a viewer has the choice of covering the components in a depth fashion or in a breadth fashion or in any combination of those. A variation in this style can be devised by creating links between components on the same level for comparison.

4.2 Parallel

In parallel architecture style, components are arranged in a two-dimensional grid. Thus, components are linked vertically and horizontally. This architecture style is appropriate for presenting different issues about more than one branch. For example, if we want to present different aspects of several programming languages. We can have components of each language such as history, syntax, range of problems domains, theoretical issues, etc. In this case, components are developed such that each component is explaining one aspect for one language. Then the components are arranged such that each components of one language are connected together and components of the same issue in all languages are connected together.



Parallel Architecture



Network (web) Architecture

The parallel architecture gives viewers the flexibility to visit components in a variety of ways. They can deduce comparative characteristics if they visit issues on various languages. They also can go through one language totally and then follow the same order in another. For an instructor, this architecture gives flexibility in covering part of the topics and gives students the chance to cover other parts on their own as independent study.

4.3 Network

The network architecture, also may be described a web architecture, though if it is called web, it may be confused with the Internet web. However, it carries the full characteristics of a web-based set of pages. Each component in the module is presented in a page. All pages are free to reference each other with no particular restriction. The only connection would be the content material. Thus a viewer may start from one pages and freely find his/her way by following links among pages.

In this architecture the instructor, or developer, has no pre-decided order for covering the information. So it is mainly up to the viewer to navigate freely within the components. More than one viewer can visit the same components in various orders each based on his/her logical thinking. On the other hand it easy to loose concentration and get confused if the purpose of study is not clear or the focus point is not clear.

4.4 Comparison

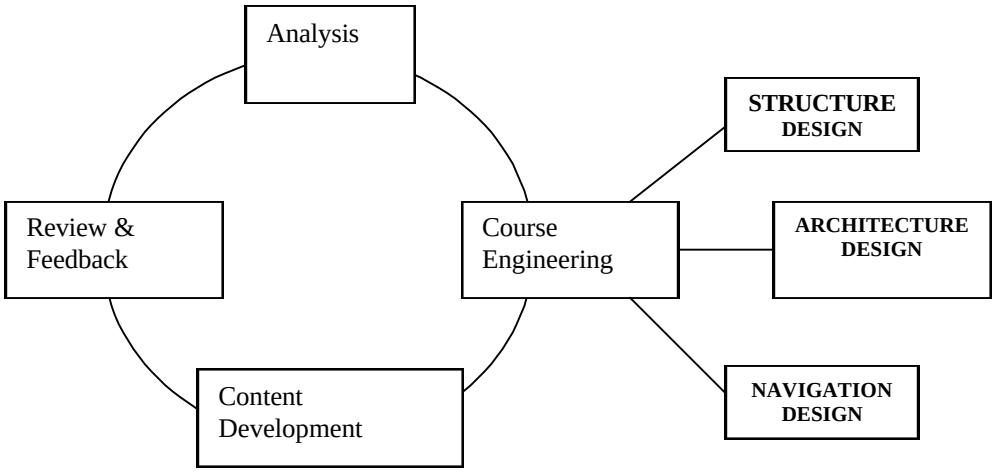
We believe that the choice of a style for the architecture of presenting material is an important part of the design. In any case, we can see that the linear architecture is strict when it comes to the ordering. The order is pre-decided and the viewer has very limited choices to go back and forth. The linear architecture is suitable if the prerequisite fulfillment is needed to understand the presented material. On the hierarchical architecture it can be viewed as a set of paths so, like the linear, it has a pre-requisite fulfillment, yet it can be navigated in a various paths. Each path though implies an ordering. The parallel architecture is very appropriate when the relationships are clear from the instructor, or logical, point of view, yet it is meant to leave for the students the freedom to navigate in their own way. As we mentioned in the example, it is very suitable for following one method and leaving for the students to, freely, investigate other methods. The network architecture has no limitations on the navigation order.

Another factor is the focus of the study. This is of particular significance in course development, in contrast to free Internet navigation. A course has objectives and it is required for every student to cover a minimum knowledge contents and skills to pass the course. There are exams and assignments that are required and each student must complete, with a degree of competence. Linear architecture, particularly if it has a conditional advancement to next components, has a focus. On the other extreme, the network architecture has no focus. The focus in this case is external and is not built in the architecture. The risk of a student losing focus or getting distracted is high. Note that one of the problems of the Internet is the distractions or loss of focus.

A third factor is the development and maintenance. The most important factor is not just the content of the components, but the links and maintaining the connections among the components. In linear architecture components are linked with one direction, thus if a new component is to be added or deleted, we have only two sets of links to be updated. On the other hand add or delete one component in network architecture, many links had to be updated.

5. Development Methodology

We suggest an evolutionary methodology. An evolutionary development lifecycle is appropriate for complex systems and it assumes that the product evolves over a period of time. Within evolutionary lifecycle models we may have incremental or spiral models. In incremental models the product is divided to several subsystems each is developed, more or less, separately, and each subsystem is integrated with other components. Spiral systems, originally introduced by Bohem 1988, is an evolutionary software process models that couples the iterative nature of prototyping with the controlled and systematic process of classical development systems. It provides the potential for rapid development of incremental versions of software. Using spiral model, the product is developed in a series of incremental releases. Early releases may be a paper model or a prototype. During later iterations, the product is more complete of the engineered product.



Course Development Methodology

In course design we assume an evolutionary lifecycle model with the activities of analysis & specification, Course Engineering, Content development, Review & maintenance. In parallel to the evolutionary lifecycle, we assume some development is needed for supporting utility, documentation, and management.

Analysis:

Analysis is the process of studying the course material and possible sources and deciding on what should be covered within the course, i.e., the depth level of each part, the method

of approaching the material, the evaluation method, etc. This output of the analysis is the course outline and sources of information. This is a process that always done in any course, however, here it is done with an automated tool and environment in mind.

Course Engineering:

Course engineering is subdivided into three tasks, structure design, architecture design and navigation design.

Structure Design:

As discussed before, structure design is the process of breaking down the course into modules and the modules into components. For online course development, the component is viewed as material to be presented in one web page. Thus, it is logically, or semantically, one coherent knowledge unit, and structurally, a page that is related to other pages. The design output is a model that includes all the pages named and each one has a clear description.

I personally, made use of software engineering notations to represent such a model. I use CRC cards to describe each component. I assume that I have one class which a module and consider all the components as instances from this class. I keep responsibility of each component and the collaboration between classes in the card. I also have a database that includes the information in the cards. I also develop a structure diagram with each component as a box, to represent relationships.

Architecture design:

For each module in the course, an architecture style is chosen. The choice is a decision on how the various components of the module will be presented. There are many factors affecting the choice, the level and objective of the module are the main factors. However, many other factors such as the available sources, the skills of students, and the relationship of the course to other courses may affect the choice. Based on the architecture, each module will be viewed as smaller components.

Navigation Design

After the decision on the structural and architecture design, we have a complete model of the course with all the components to the smallest level. However, as a student will study the material he/she has to navigate this structure gathering information and gaining knowledge. As part of the design process, or may be as an evaluation for the design, we have to review the navigation process imagining that a student has to visit the various pages and logically gaining information based on the enquiries in his/her mind about the topic [1]. This process is an important part of the design and is not an easy task.

To achieve this task, we may employ “Use Case” and scenarios. We have to define various types of students in various situations. For example, a student starting to

know about the course, a student in the middle of the course, a student finishing one module, etc. We also, should consider a situation in which a student is studying one part and wants to review a previous particular concept, not the whole component. Another situation is a student who wants to review material just before an exam. All those situations are viewed on the decided model to see if the suggested design provides support for all of them.

For each one of those situations, we try to find components and count the navigation links and how could it serve the student need. The purpose of this task is twofold. First it is a review on the design model. We can look at the design from student's point of view. In fact it views various students in various situations to verify that the design provides what is needed. Second, it would provide different starting points for different purposes. In many of the current situations a student is given a single starting point and in other it gives several stating points but it scrolls in the same order if a student needs it or not. In web applications, one of the measures of satisfying the user needs is the count of clicks from the starting point to reaching the wanted goal.

Content Development

This process is to be achieved after the engineering has been completed. The output of the engineering is the model of the course material, including components, each with a clear definition of its role within the course, the structures, i.e., the relationships of all the components and their interaction with each other to provide the body of knowledge. Thus, the next step is to fill in all the components with the required material.

The content is not meant to be only one part or one type. Content can be in one or more media. It could be text, images, illustrations, audio, or video material. It is not necessary even in one media; it could be a mixture of media. Also, the content is not just the explanation of the topic, but it could be also supporting material such as examples, case studies, actual practices, etc. In addition, we can consider tests either self-tests that are used by students or exam questions that are to be used to evaluate students' performance. However, I suggest that a component has only one type of media and one purpose. So in deciding on the components and structure, this should be taken into consideration. However, it may happen that as we go through component development, we discover that we need to add more components to the structure. In this case the model is to be reviewed. The output of the content development is the complete course material structured and ready to be delivered.

Review and Feedback:

As the content material development is complete, the system is built and ready to be used by students. The process of review starts from this point. Initial reviews are needed before the actual having the course used by students. One part of the review would be reviewing the content, each component against its stated objectives. Also the supporting material has to be reviewed to assess its consistency with the body of the components. Questions and cases also have to be reviewed against the material itself.

Second is the review of flow of information within the course material across components. This is not an easy task, since based on the structure and architecture of the modules, may have more than one possible path. Thus, a review of the flow within each path should be planned and achieved.

After reviewing the material and the course is provided to students, we still have to follow on reviewing the material and other supporting material. Meanwhile we may receive feedback from students on either the content or the structure and flow of knowledge. At this stage, we keep all the comments and feedback with suggested solutions. As the course is concluded, we should analyze all the comments and the new additions to the field and come up with a new engineering for the course. In the second path, we do not have to change the whole course. We can see the required changes and plan for it, come up with an updated model. The changed components would be the only changes that are required. We can reuse the components that are not changed.

6. Management Issues

Online course offering is not just the display of information or the presentation of knowledge. There are some additional issues that need to be considered; we call them management issues. They are categorized in three areas, student record management, course configuration management, and shared knowledge management.

Student Record Management

There is a need to keep track on what modules and components that each student have completed. This is necessary if we assume asynchronous setting in which each student is visiting the online material independently. It may be somewhat necessary if the online material as a support material for extra reading. Then it would be only preferable in synchronous setting in which the material is arranged to be going exactly with the class material. In this case the site would be useful as a feedback and record keeping for future improvement rather than a student record.

Most of the existing systems provide the whole table of content to the student and he/she would start at a given point. This is reasonable for the supporting material, but even as a support material it does not provide any information about how the material has been visited or how students made use of it. Thus, a management module is part of an educational platform. It acts as part of the educational process of any material. We assume that it simply keeps students' names that are allowed to use the presented material. Then as a student view the course material, the management system keeps track on what modules have been viewed by each student.

Moreover, if the progress of the student within the course is conditional, i.e., there is a test at the end of each module and a student may advance from one module to another if he/she passes the test, the management module would be responsible for keeping track on students' scores. This is more important if the course is offered on asynchronous basis.

In my personal opinion, the management of student records will grow in importance, as course offering will advance. It may also be used in systems where the course is offered as training material for on-job training. In this case we need to keep the extent of coverage of material both in quality and quantity.

Configuration Management

Another issue that is well shared between software engineering and course offering is configuration management. In software product this is mainly concerned with management of the components as they change either for correcting errors, feature enhancement, or for improving the performance. In course engineering, we want to emphasize the reuse of material. Thus, from one offering to another, course material is reviewed and redeveloped for the new course. However, this engineering approach tries to make use of the old course material to develop the new one.

We always assume that educational material would be continuously improving; yet there is no need to redevelop everything from scratch. We can improve the course by adding new components to the old ones, giving more choices to students. We may also change the presentation without changing the content. We may add more examples or cases. If exams are included in the system, new questions need to be added to the system. We may certainly add a completely new module.

Having a well-documented design and a good meta-data about the modules and the components, supports, easier and more manageable changes for the course material. First, the class diagrams and CRC cards provide a good support for relationships among content material in the course. Thus, if the content of one component or a module needs to be changed, the design documentation provides a map of related components that need to be reviewed to maintain consistency of the material. We suggest that this part should be handled in another module that is course independent. This would be like a metadata catalogue of the modules and components. This database would respond to queries as: given a particular component, what are the other components that are related to this one. Thus, this would keep track on the module structure and the component structure. This, if built carefully, would substitute the CRC cards, since it would represent the information on the structure during the development. Then as the course is fully developed, it will be the base for improvement and substitution of components as the course is offered for another time.

Glossary and Domain Knowledge

As a course is being offered, a student may need to refer to supporting material that is not within the scope of the course, but it is needed as a reference to gain the required knowledge in the course. This may be material from prerequisite courses or from other courses such as collateral requirement from various disciplines. However, what concerns us here is material such as the glossary of the terminology or some general knowledge that is related to understanding the material.

On the design level, the structure of the course would not include this material though it may be needed for some students. It is particularly important in technical course where the terminology have to be emphasized. Again, this type of information is not owned by one particular course and is needed for several courses. It could be domain dependent rather than course dependent. If that is the case, the handling of the glossary is assumed to be free from any module and it should get back to the same module. Thus, it should not change the architecture of the module.

Ideally, if we imagine that all courses in one branch of science already have migrated to the online setting, this would exist in no extra effort. An instructor would simply refer to a module or material out of the design and yet can go back and forth from his material. Also the foundation of the branch is shared by all the courses. Thus, this feature would eventually be solved at no extra cost. However, as we are designing the course, we have to consider providing this support as part of the course development even if it is not part of the course material. In fact, sometimes we would provide it even if it is already there if we need only a summary or a subset from the knowledge. The objective at this point would be not to distract the student.

7. Conclusion

We assume that online course offering has different characteristics than classical courses and to handle it we need to engineer a methodology for course development. In this paper we tried to define a methodology based on analogy with software products and web applications. We assume engineering of an online course involves structural design, architectural design, and navigation design. The design is preceded by analysis and succeeded by content development. We also assume that the methodology is evolutionary and spiral to make use of one offering in a process of continuous improvement of course material. However, we see a need to add some of the management requirements, which we plan for a future work. Currently this methodology is used to plan and develop a course in software engineering and hopefully in the future would mature to be used in a generic basis.

8. References

[1] H. Azzam, D. McBrin, and S. Meredeth, “On Web Site Accessibility,” Proceedings of 6th International Conference on Software engineering and Applications, Nov 2002.

[2] Dennis, B, Wixom, and D. Tegarden, Systems Analysis and Design: An Object Oriented Approach,” John Wiley, 2002.

[3] J. Peters and W. Pedrycz, “Software Engineering: An Engineering Approach,” John Wiley, 2000.

[4] R. Pressman, “Software Engineering: A practitioners Approach, 5e,” McGraw Hill, 2001.