Leena Hiltunen (leena.hiltunen@mit.jyu.fi)

Tommi Kärkkäinen (tommi.karkkainen@mit.jyu.fi)

TOPIC-CASE DRIVEN APPROACH FOR WEB-COURSE DESIGN

InBCT 2.4

Agora Learning Laboratory

21.8.2003

Agora Center

Keywords: topic-cases, web-course design, online learning, reusable learning objects

Abstract: A topic-case driven methodology for web-course design and realization process is introduced. The proposed approach is based on software engineering metaphors for capturing the necessary steps for creating web-courses using a content-based development method. Also, pedagogical and technical activities as well as unitwise and overall assessment are considered. Finally, possibilities for constructing and maintaining web-course repository and corresponding training programs are discussed.

Tiivistelmä: Tässä raportissa esitellään aihetapauslähtöinen tuotantoprosessi verkkokurssien suunnittelua ja toteutusta varten. Kehitetty menetelmä perustuu ohjelmistotekniikan metaforien käyttöön tarvittavien suunnitteluvaiheiden muodostamiseksi, joiden avulla verkkokursseja voidaan toteuttaa selkeästi strukturoitua ja sisältölähtöistä prosessia noudattaen. Työssä tarkastellaan myös pedagogista ja teknistä suunnittelua sekä yksikkökohtaista ja yleistä arviointia. Lopuksi tarkastellaan mahdollisuuksia verkkokurssiarkistojen sekä koulutusohjelmien suunnittelua ja ylläpitoa varten.

Contents

1 Introduction

The creation of digital content is regarded as the next wave in the development of the information society (e.g., Finnish Ministry of Education, 2002; Council of European Union, 2000). At the core of content production – independent of the purpose of the material to be produced – one should employ a content creation and development process, which, at its best, supports structural and incremental development and, thus, also reusability of the resulting material as suitable learning objects (Jacobsen, 2001; Catenazzi & Sommaruga, 2002; LOM, 2002). The purpose of the present work is to describe such a development process for web-course design.

Even nowadays web-courses are far too often simply based on exporting traditional written course materials to the web without proper planning and pedagogical design. Technical decisions may already have been made by someone else if a particular platform is already in use in your organization. Usually the testing and evaluation phases of the web-course are already completed and we are simply relieved to have something up and running. This monolithic approach makes the extension and reuse of a web-course (or parts of it) difficult.

There are no unified practices for web-course design. There exists, though, some reported experiments and trials related to some parts of individual courses. One can also find some design process descriptions for web-course creation that are mostly related to software engineering (e.g., White, 2000; Montilva, 2000). However, all of the existing methodology fails to describe the development process that would allow for the well-managed integration and incorporation of pedagogical knowledge, communication and cognitive tools. (Multisilta, 1997). Moreover, an article by Pekkola (2003) describes CSCW (Computer Supported Cooperative Work) of learning groups as also based on (pieces of) documents for further elaboration.

Humphrey (1998) emphasizes effective planning and quality management in software engineering. He also highlights the use of improved methods. Each of these, we believe, is a useful principle in web-course design. Hoover (1998) applies John's model for human-computer interaction (John et al., 1992) in their software engineering based TAP-D model, where, "software developers apply domain knowledge, computing theory, and software development techniques to specify, design, and evaluate software for a particular application." This same model can also be applied in web-course design because, not only does content-based knowledge but also pedagogical theories and practice (e.g., authentic assignments) influence the design of a web-course during the learning process. Moreover, in Boehm et al. (1998), an extension of the popular Spiral Model of software development has been used to design, implement, evaluate and improve software engineering core courses for the USC MS-degree program.

The content of a web-based course is similar to the functionality of a computer program: they both driver further development, presenting functionality and contents in the best possible way to all users and students to enhance usage and learning. In software

engineering (SE) a structured way for presenting the general functionality of an application to be implemented are the use cases (Jacobson et al., 1992). Together with the use case diagram they capture and present in a hierarchical way the so-called functional requirements of a software system. Hence, we use the use case as key SE-metaphor behind the introduction of a basic element of contents: *topic-case*. In our approach the topic-cases carry the whole web-course development process from the initial topics and supporting material through pedagogical and technical considerations into the final realization and assessment. In particular, the proposed technique naturally supports utilization of large, possibly distributed team of domain experts for creating the key contents.

The contents of this work are the following: First, in Section 2 we summarize the necessary preliminaries for the rest of the paper. In Section 3, we introduce the topic-case driven web-course development process on the general level and in Sections 4-7 we describe the activities needed during different phases in more detail. In the last two sections we discuss the assessment, realization and integration of different web-courses created using the proposed design process.

2 Preliminaries

A general process definition for software design usually consists of three elements: phases, activities and tasks. A phase represents the highest level of abstraction. Each phase contains a logically grouped set of activities and tasks that perform a process development function. Each phase must be passed in order to realize the entire process.

Each phase can be divided into activities that are composed of tasks. Activities can overlap one another when tasks are related to each other by a certain function or certain participator.

A task represents a particular set of steps that occur within an activity. It is the lower level unit of the whole process with detailed information for completing the activity itself. Some tasks are performed only once during the entire project lifetime and others are performed for each iteration release. Typical task descriptions include:

- Management responsibilities and other roles; who is doing what and when
- A brief explanation regarding why each task is performed
- Possible references that describe the required processes to be performed or the documentation to be produced
- Inputs (possible documents, data, or other products) that are required for or used during this task, or standards that must be adhered to in completing the task
- Procedures or steps that must be performed within a task
- Outputs (documents, data or other products) that are produced during the task

Many benefits of the utilization of well-defined development methods have been established in information systems design. For example, Smolander et al. (1990) list their findings concerning methods as follows:

- Enhanced standardization of documentation and system work
- Makes system development easier and faster
- Ensures better application quality
- Structures system work, thus making project management easier
- Improves maintainability of applications
- Yields less dependency on key persons
- Allows for easier construction of large databases
- Makes testing easier
- Avoids naming problems

Recently White (2000) and Montilva (2000) described development processes for web-course design and we briefly review these two approaches.

2.1 A Web-Course Development Process by White

White (2000) describes the web-course development process used at the University of Houston Clear Lake (UHCL) for creating selected courses in SE. The process is divided

into three concurrent sub-processes: Standards and Policy Creation, Course Material Creation and Web-Site/Web-Page Creation. These sub-processes can be treated separately, but this requires a more evolutionary style of development. In each sub-process, key process players (actors) and major results/documents that must be produced have been introduced. White's model reminds one of a modified waterfall model for software development (e.g., McConnell, 1996).

In Standards and Policy Creation sub-processes, needed regulations and constraints for different academic actors are created. The other two sub-processes proceed concurrently with each other, resulting in the final web-based course. White discusses this concurrence with in use of WebCT: "If one is to use such a tool one must begin the web design at the same time as course material design in order to work most efficiently and productively (that is, to avoid re-design and re-implementation of possibly major portions of the course)."

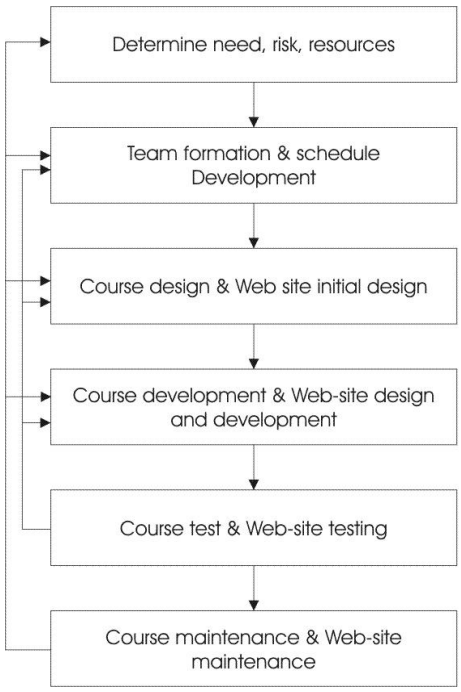


Figure 1. Major activities of the web-course development process by White (2000).

White uses software engineering metaphors when she describes the major activities of the web-course development process. In some stages (see Figure 1), there is more than one concurrent activity occurring during the same stage. Also, exact documentation during all stages is applied according to general practices in software engineering processes.

Next we describe the actual activities within the development steps. At the first stage, all risks and benefits that the online course might produce are analyzed by the Dean and

faculty. Next, all the needed resources (hardware, software and people support) are considered, once the courses to be provided are selected.

In the second stage, courses are assigned to content experts to develop the course materials, and to form the development teams. A schedule for development of the courses is also planned. The third stage is the actual design phase, and includes the creation of the course syllabus, course policy, course objectives and content design. Content design is restricted to weekly unit overviews (weeks topic, objectives and major assessments). The web developer designs the top-level web structure of the course based on design documents produced by the content expert and instructional designer. This structure contains material and communication mechanisms (e.g., chat rooms and bulletin boards) as part of the initial web site design.

During the fourth stage the actual course content is created and finalized. All assignments, student guides, supporting materials, etc. must be created and converted to an appropriate format (html, pdf, etc.). Fifth, a testing stage follows after the course has been fully developed and becomes available online. In testing, students answer questions about the materials provided, conducting some assessments and tests attempting to determine the overall success of the course, including its strengths and weaknesses. According to White, "the idea of the test is to determine the weak points and correct them before offering the course at large." Later, web-courses will require maintenance support on the website. This last stage was still under discussion at the UHCL.

By fall 2000, this process was used for design, development and testing of three strictly web-based software engineering courses. Newer information was not available on the UHCL web site (see http://sce.cl.uh.edu/swen/index.htm).

2.2 A Software Engineering approach by Montilva

Montilva (2000) describes "a method that applies object-oriented software engineering to the process of developing web-based courses." Montilva describes the phases, steps, activities, and techniques as including:

- Analyze and specify the technical and instructional requirements of a course
- Design the structure, interface, content, and interaction of the course
- Produce the content, user interface and media required by the course
- Deliver the web-based course to its users

Montilva's six-phase method (Figure 2) has been used to develop web-based study guides for distance education. The method begins with an analysis of web-course domain and iterates over the entire development cycle ending with its delivery. The evaluation phase (verification & validation) has a central role, meaning that evaluation of results begins in the first phase, instead of being executed at the end. This model reminds one of the Star Model of Preece et al. (1994) for human-centered software development.

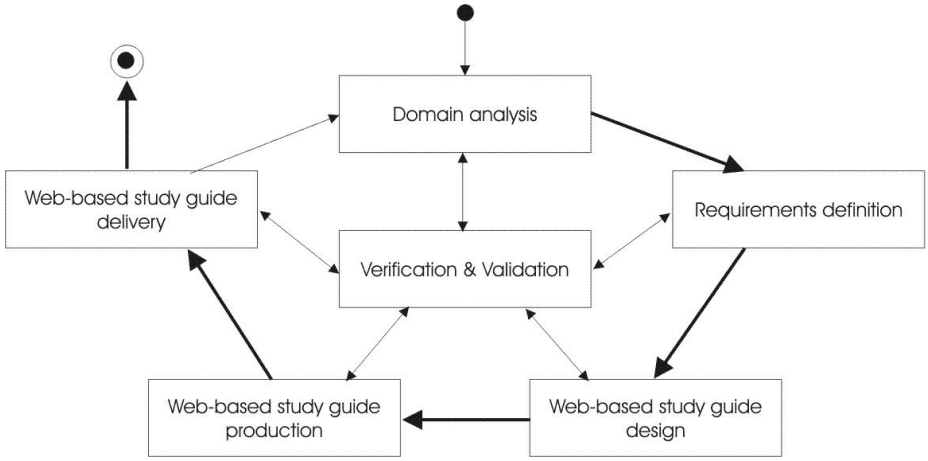


Figure 2. The phases of the method by Montilva (2000).

During the first phase an analysis of the web-course domain is performed including:

- Identification and analysis of the subject of the course, organization of the content in themes, and definition of the objectives and goals of the course
- Assessment of the student's prior knowledge on the subject, skills required before taking the course, motivation, and abilities yet needed to achieve computer proficiency, follow a distance learning course and conduct independent study
- Analyzing the instructor's abilities: subject-matter knowledge, distance teaching experience and attitude, computer proficiency, knowledge and experience on Internet services (WWW, FTP, E-Mail, News, etc.), and pedagogical profile
- The learning environment: location of the students, the telecommunication technologies and hardware-software platform, the social and physical environment, and time availability for completing the course.

In the second phase, all requirements that should be satisfied by the final course are defined and specified. Through requirements definition the development team considers the most important features during the design phase, and verifies and validates the study guide once it has been designed and produced.

Requirements include:

- Learning activities that students should perform (reading, writing, viewing, listening, group interaction, testing, etc.), and the length of the course in weeks and number of study hours in order to produce a timetable for the course
- Interaction requirements: types of interaction to be supported and other media to be used together with the Web study guide
- Development and operational resources: time, hardware, software, people and financial support, to estimate the time and cost of developing a Web study guide
- Quality attributes including structural attributes (modularity, visibility, balance, modifiability, navigation), interface attributes (organization and visualization of

hyperlinks and multimedia items by page length, background color and texture, design grids, size and resolution of graphics and images, and typographic design) and content attributes (the scope of the content, the logical sequence and organization of the content, its completeness, the way of stimulating or motivating the student, the feedback on assignments, the method used for evaluation of the content and the repetition and summary of the most important ideas), to achieve well established Web style rules and design criteria

The third phase includes designing the web study guide, focusing on different aspects of design such as structural, navigational, conceptual and sensorial aspects. This proceeds in the following order:

- Design of the basic structure: main page, units, lessons, themes, etc.
- Design of units and lessons including the structure of each unit and linking of the learning activities
- Design of web pages by modeling the structure and behavior of each page, and designing the items of each page
- Building a prototype based on the design specifications
- Verifying and validating the design by using the prototype that should satisfy all of the requirements

Production of the Web study guide during the fourth phase includes

- Producing the multimedia items, including animations, images, and audio and video clips
- Assembling the items into the prototype
- Verifying and validating the Web study guide – a final evaluation of requirements fulfillment by developers and testing by real students

The final, fifth phase is delivery. After the Web study guide has been stored on the Web server, it will be accessed by the remote students using a Web browser. This phase concludes the development process and begins the maintenance stage.

According to Montilva, “one of the most important features of the method is its emphasis on the quality of the product.” The method is specific to Web study guides, and it covers the whole life cycle. Besides, the verification and validation process is used as, “a continuous activity that have to be performed through all phases of the method.”

2.3 The Unified Process

Building a web-course is similar to the design and implementation of a software application. Hence, terms from software processes that form the basis of software development, such as feasibility study, analysis, architecture, design, implementation, testing, iterative and incremental can also serve as a well-established conceptual framework for web-course design. Because the initial stages of our web-course development method mimic the Unified Process (UP), we make a short review of this approach next (with some direct quotations) based on Jacobson et al. (1999).

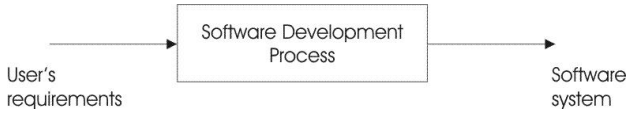


Figure 3. A software development process by Jacobson et al. (1999).

The Unified Software Development Process is, like all other process models, a development process where the set of activities needed to transform the user's requirements into a software system are organized (see Figure 3). The Unified Process is use-case driven, architecture-centric, iterative, and incremental. The goal of the whole process is "to guide developers in efficiently implementing and deploying systems that meet customers needs."

Software systems should be designed to serve its users, so we must know what its prospective users want and need. The user could be a human or another system that interacts with our system. In response to the user's actions (e.g., pushing a button or clicking the mouse) the system performs a sequence of actions that leads to a response. Jacobson et al. describes this sort of interaction as a use case, "a piece of functionality that gives a user a result of value." Moreover, "all the use cases together make up the use-case model which describes the complete functionality of the system," by capturing all functional requirements.

The use-case model answers the question: What is the system supposed to do? We should think about the value of the functions to users, and not just speculate as to what functions might be desirable. With use cases we can find the true requirements and represent them in a suitable way for users, customers, and developers. Use cases are not just tools to capture all the requirements of a system. They also drive its design, implementation, and test when developers create design and implementation models that realize the use cases.

The Unified Process is use-case driven, but the system architecture (i.e., general structure of software) establishes the skeleton for technical design. The architecture is illustrated using different views of the system being built; it is a view of the whole design with the important characteristics made more visible by leaving details aside. "Process helps the architect to focus on the right goals, such as understandability, resilience to future changes, and reuse."

Usually software projects are large and continue over several months or a year or even more. This is one reason why projects are usually divided into smaller mini-projects. "Each mini-project is an iteration that results in an increment. Iterations refer to steps in the workflow, and an increment, to growth in the product." According to Jacobson et al., the selection of what is to be implemented during the current iteration is usually based on two factors:

- The iteration deals with a group of use cases that together extend the usability of the product as developed so far.
- The iteration deals with the most important risks.

"In every iteration, the developers identify and specify the relevant use cases; create a design of the chosen architecture as a guide, implement the design in components, and verify that the components satisfy the use cases. If an iteration meets its goals, developers proceed with the next iteration. When an iteration does not meet its goals, the developers must revisit their previous decisions and try a new approach."

2.4 Background on pedagogical design

The basis of design of all educational and learning environments should be some kind of model or method regarding learning and teaching (Manninen & Pesonen, 2001). Mostly only models and methods based on constructivism have been used with new learning environments, but one should remember that the new technologies also enable new kinds of educational methods. Manninen and Pesonen remind us also that before we can take full advantage of these new educational methods, we should be able to recognize and adapt basic rules of learning.

Mezirow (1981) presents three generic domains of adult learning:

- *Instrumental learning:* the objective is to increase empirical knowledge and "technical rules" on how predictions about observable events can be proved correct or incorrect and what is or is not an appropriate action; e.g., a student learn to use the computer
- *Communicative learning:* the objective is to increase knowledge on binding consensual norms that define reciprocal expectations about behavior and that must be understood and recognized; e.g., learning in negotiations, instructional situations and co-operation
- *Emancipatory learning:* the objective is to increase the interest and knowledge in self-knowledge, self-reflection and self-awareness

After being aware of these generic domains it is possible to design learning environments where different kinds of learning styles can be supported with communication tools, cognitive tools and learning materials.

It is also important to realize that different didactical and pedagogical approaches fit into different learning situations and needs. There are many suitable approaches for virtual learning environments, e.g. (Manninen & Pesonen, 2001):

- *Instructional learning:* focus on teaching and motivation; teacher-centered, predetermined goals, contents, teaching and assessment methods; responses to impulses, receiving of knowledge
- *Cognitive learning:* focus on complete learning; processing new information and connecting it into existing data structures

- *Constructivism:* focus on construction of knowledge; commitment to learning, independence, situational and contextual involvement, peaking of activity
- *Humanistic learning:* focus on self-improvement; problem-centered, self-directed, highlight on students own responsibility
- *Critical humanism:* focus on consciousness; awareness of characteristic values, attitudes and types of action.

When we move from the traditional classroom onto the web, the teacher's role is definitely no longer just to deliver information. The teacher is now helping students to deal with new information and management of knowledge. Responsibility for learning is transferred from teacher to students, but the teacher's role as an instructor is remarkable. Tella et al. (2001) lists five key roles of the teacher on the Web:

- *Motivator:* keeps students' motivation and activity at a high level by focusing attention on students, by offering proper learning materials, and by maintaining collaboration and co-operation. The teacher is asking, demanding, inspiring and persuading students to participate. The teacher speaks out and responds to students' activities, pays attention to the students, creates learning opportunities, and motivates students by his or her own actions. Personalized feedback is also very essential in web-based learning.
- *Networker:* establishes networked relations to different experts and specialists and offers these resources also for students use.
- *Organizer:* organizes teaching and learning environments that drive students into collaborative learning by making choices between different tools, applications and media. The teacher organizes, structures and sets the rhythm for the course, sets goals, conducts the course based on the flexible study plan, makes stimulating questions, and comments and guides the discussion.
- *Signaler:* creates nets of communication, informs and guides students during the learning process by making specific instructions and guiding questions on the web. The teacher creates the rules for communication and ensures that all students will understand them.
- *Instructor or tutor:* makes it possible for students to learn better, but without controlling too much. The teacher helps students to understand, guides them toward active learning, and enables the process where the student internalizes the external knowledge and transforms it into his or her own knowledge.

There are also other roles for teachers, such as assessor, supporter, expert or storyteller. However, the teacher needs the same kind of didactical and pedagogical skills as in the traditional classroom, but the form of teaching and the teaching environment are changing. Furthermore, teachers as well as students need new computing and communication skills.

2.5 Technical division of virtual learning environments

Technically, virtual learning environments (VLE) can be divided into three parts: cognitive tools, communication tools and hypermedia-based learning material (Multisilta, 1997). VLE can be composed of one or more of these parts. Moreover, technical design might not follow this division because one technical solution can represent one or more of these

parts. As described in the previous section, modern pedagogics deals a lot with learning and interaction support, aspects of design that are enabled through these tools.

Cognitive tools (or problem solving tools) guide and expand the thinking and learning processes of the student (Häkkinen, 1996; Multisilta, 1997). Lajolie (1993) describes four categories of cognitive tools on the basis of how computers can be used in different situations:

- *Tools that support cognitive processes,* e.g., memory and metacognition
- *Tools that share the cognitive load,* e.g., the computer carries out lower level tasks for learner thus freeing up attentional resources to accomplish higher order thinking skills
- *Tools that assist the learner to engage in out-of-reach activities,* e.g., by providing simulations with safe opportunities or without physical limitations from the real world
- *Tools that provide support for hypothesis testing,* e.g., by providing multiple hypothesis paths with support or coaching in the context of such hypotheses.

Cognitive tools can also be categorized from other perspectives. Jonassen (1992) lists three dimensions of cognitive tools:

- *The dimension of control:* concerning where the control over the learning situation and the artifact is, ranging from total teacher control to total learner control
- *The dimension of generativity:* concerning the view of learning and knowledge, permeating the learning situation, ranging from pure presentation to genuine creation
- *The dimension of engagement:* concerning the way learners act in the learning situation, ranging from passive to active.

Communication tools enable communication and cooperation between users in VLE (Multisilta, 1997). Examples of communication tools are E-Mail, discussion groups, chat and video-conferencing. Communication tools can be categorized, e.g., from three different perspectives:

- *Time linkage:* asynchronous (different time) or synchronous (concurrent)
- *Direction:* unidirectional (e.g., bulletin board) or bi-directional (e.g., E-Mail)
- *Size of target group:* one-to-one, one-to-many, many-to-one or many-to-many.

2.6 Usability and pedagogical usability

In web-course design, as in software development nowadays, usability is one of the key issues. Usability has been defined by Jacob Nielsen, Brian Shackel, and in ISO 9241-11:1998 standard. Here we only consider Nielsen's definition.

Nielsen (1993) defines usability with five attributes:

- *Learnability* – the system should be easy to learn

- *Efficiency* – the system should be efficient to use
- *Memorability* – the system should be easy to remember
- *Errors* – the system should have a low error rate, and if user make errors he or she can easily recover from them
- *Satisfaction* – the system should be pleasant to use

In learning systems, pedagogical issues, such as support for learning, are important. Jonassen (1995) has identified meaningful learning as having the following qualities:

- *Active* - learners are engaged by the learning process in the mindful processing of information, where they are responsible for the result
- *Constructive* - learners accommodate new ideas into prior knowledge
- *Collaborative* - learners work in learning and knowledge building communities, exploiting each other's skills while providing social support and modeling and observing the contributions of each member
- *Intentional* - learners are actively and willfully trying to achieve a cognitive objective
- *Conversational* - learning is inherently a social, dialogical process in which learners benefit most from being part of knowledge building communities both in class and outside the school
- *Contextualized* - learning tasks are situated in some meaningful real-world task, or are simulated through some case-based or problem-based learning environment
- *Reflective* - learners articulate what they have learned and reflect on the processes and decisions that were entailed by the process.

Horila et al. (2002) combined Nielsen's usability attributes with Jonassen's qualities and defined the criteria of pedagogical usability for digital learning environments. In their approach, pedagogical usability consists of the following 11 concepts:

- *Learnability* – is it easy to learn to use the system?
- *Graphics and layout* – how different pictures and figures have been joined into other elements of the system?
- *Ease of use: technical and pedagogical approach* – can the user use the system independently? What kinds of support processes are needed?
- *Motivation* – how motivating the system and its content is?
- *Suitability for different learners and different situations* – how well are different learning situations and learning styles supported by the system?
- *Technical requirements* – is there enough computers available for students, do teachers and students have the proper equipment, and does the system work in a stable way?
- *Sociality* – is the system designed for individual learning or is the social activity between teacher and students considered in the system?
- *Interactivity* – is there some interactivity included into the system?
- *Objectiveness* – is the system target-oriented?
- *Added value for teaching* – how beneficial is the system by means of all the work that its use requires?
- *Intuitive efficiency (teacher, student)* – are the users willing to use the system, how effective is it from the users point of view?

Pedagogical usability is a concept that forms a kind of bridge between pedagogical and technical design of VLEs. The given attributes are also useful in the assessment process, although it seems that some parts of the proposed definition by Horila et al. are not in the same level of abstraction than the other parts or the underlying theories by Nielsen and Jonassen. More precisely, *Technical requirements* and *Graphics and layout* are technical prerequisites or means to fulfill the other, more general attributes.

2.7 Human-centered design

Because learning should be learner-centered, learning systems should be also. That is why the learner's requirements and activity should be the main objectives in designing learning systems. ISO 13407:1999 standard identifies four main activities of human-centered design for interactive systems (see Figure 4):

- Understand and specify context of use
- Specify the user and organizational requirements
- Produce design solutions
- Evaluate designs against requirements

According to ISO 13407:1999 standard, we should in the design phase:

- Use existing knowledge to develop design proposals with multi-disciplinary input
- Make the design solutions more concrete using simulations, models, mock-ups, etc.
- Present the design solutions to users and allow them to perform tasks (or simulated tasks)
- Alter the design in response to the user feedback and iterate this process (see Figure 4) until the human-centered design goals are met
- Manage the iteration of design solutions

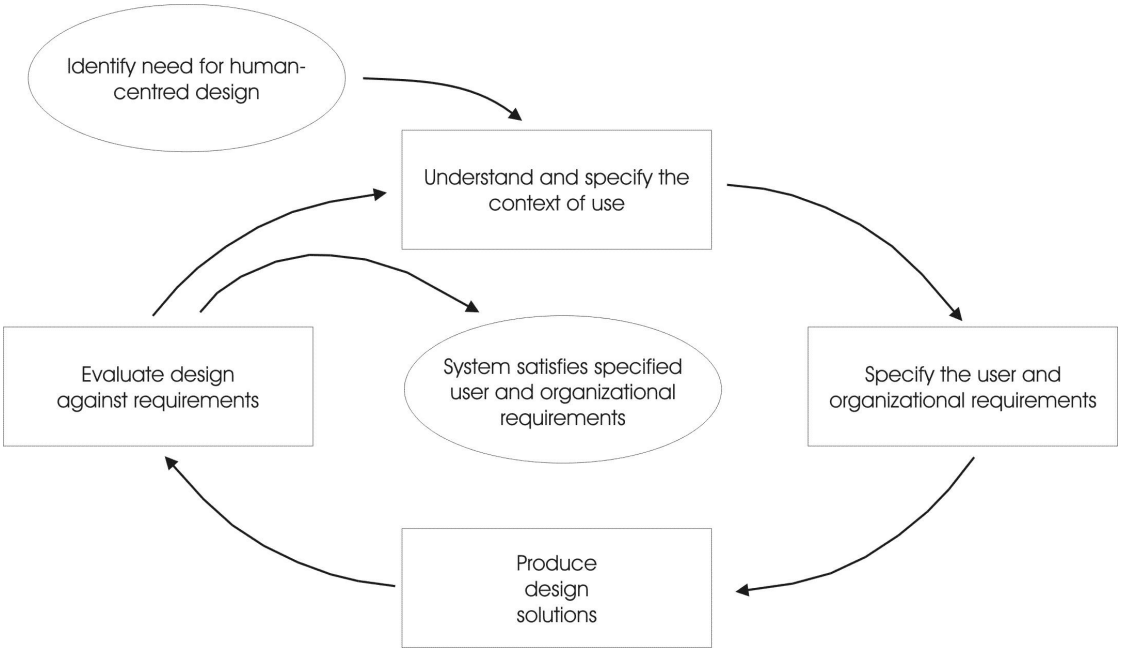


Figure 4. The interdependence of human-centered design activities (ISO 13407:1999).

2.8 User interface design as a part of technical design

Nielsen (2000) introduces some issues that should be remembered during web page design:

- "The user fundamentally controls his or her navigation through the pages. The user can take paths that were never intended by the designer." Therefore, "designers need to accommodate and support user-controlled navigation."
- "We have to design pages that will work (also) on small screens … resolution-independent pages that adapt to whatever size screen they are displayed on." So, "never use a fixed pixel-width for any tables, frames, or other design elements … Instead of using fixed sizes, you should specify layouts as percentages of the available space."
- "It is recommended to separate meaning and presentation, and to use style sheets to specify presentation, but doing so works better for informal content than for interaction"
- "Avoid non-standard codes if possible and, if not, at least use reasonable defaults that will work"
- "The only format you can use with complete confidence is the original HTML 1.0 specification. Anything beyond that will be beyond the capabilities of some of your visitors (users)"
- "Fast response times are the most important design criterion for web pages"
- "Web pages must be designed with speed in mind … To keep page sizes small, graphics should be kept to a minimum, and multimedia effects should be used only

when they truly add to the user's understanding of the information." (cf. Section 2.6). This does not mean that we should design only boring web pages, "style sheets can be used to improve page design without incurring a download penalty."

- "The top of the page should be meaningful even when no images have been downloaded" and "the browser must draw the top of the page quickly."
- "Use ALT text attributes for images so that users can understand what they are"
- "Cut down on the complexity of your tables, split the information into several tables"
- "Links are the most important part of hypertext", but links "should not be overly long" and "only the most important information-carrying terms should be made into hypertext links". So, avoid links like "Click here".
- Links should include short explanation (a link title) of the target of the link
- "It is better to link a small number of highly relevant external pages than to link to all possible alternative sites on the Web"
- "Frames: Just Say No":
    - "Many browsers cannot print framed pages appropriately"
    - "Frames are so hard to learn that many page authors write buggy code when they try to use them"
    - "Search engines have troubles with frames"
    - "Some browsers make it difficult to bookmark frames"
    - "Most users prefer frames-free designs"
- "Provide printable versions of any long documents" and make sure that all pages are printable (most of the users want to print long documents to read offline)

3 A topic-case driven development process

Key questions for web-course design are how to design learning material that benefits from using the web and how and when to integrate such a (web-) pedagogic into training that enhances learning. Although being important steps toward a structured method in which to develop web-courses, we feel that these two central aspects are not clearly captured in the existing approaches, e.g., by White and Montilva. These and other existing processes seem to be mainly organization-centric (time & schedule drive the development of contents that is immediately organized, e.g., on weekly units) and not as much learning-centric.

In our approach, we also utilize metaphors from software engineering to describe a unified way to design and realize web-courses. Jacobson et al. (1999: xviii) describe a SE process as "who is doing what when and how to reach a certain goal … an effective process provides guidelines for the effective development of quality software. It captures and presents the best practices that current state of the art permits." Moreover, the process guides all the participants involved and leads to more stable development steps. All these issues also fit into web-course design.

In general, our web-course design and realization process contains five phases: background study, content design, pedagogical design, technical design, and realization and assessment (see Figure 5). The proposed approach allows incremental and iterative development of the web-course (again following UP). Moreover, it can be utilized as a content development miniproject within other similar methods, e.g., by White and Montilva. In the next sections we depict each phase with more details using general activity and phase product descriptions. We do not define exact tasks, management responsibilities and precise roles in each activity, because they all depend much on organizational issues and the actual environment for carrying out the development project. Moreover, due to the same reason we also leave aside all kind of metadata that could and should be documented as part of the development.

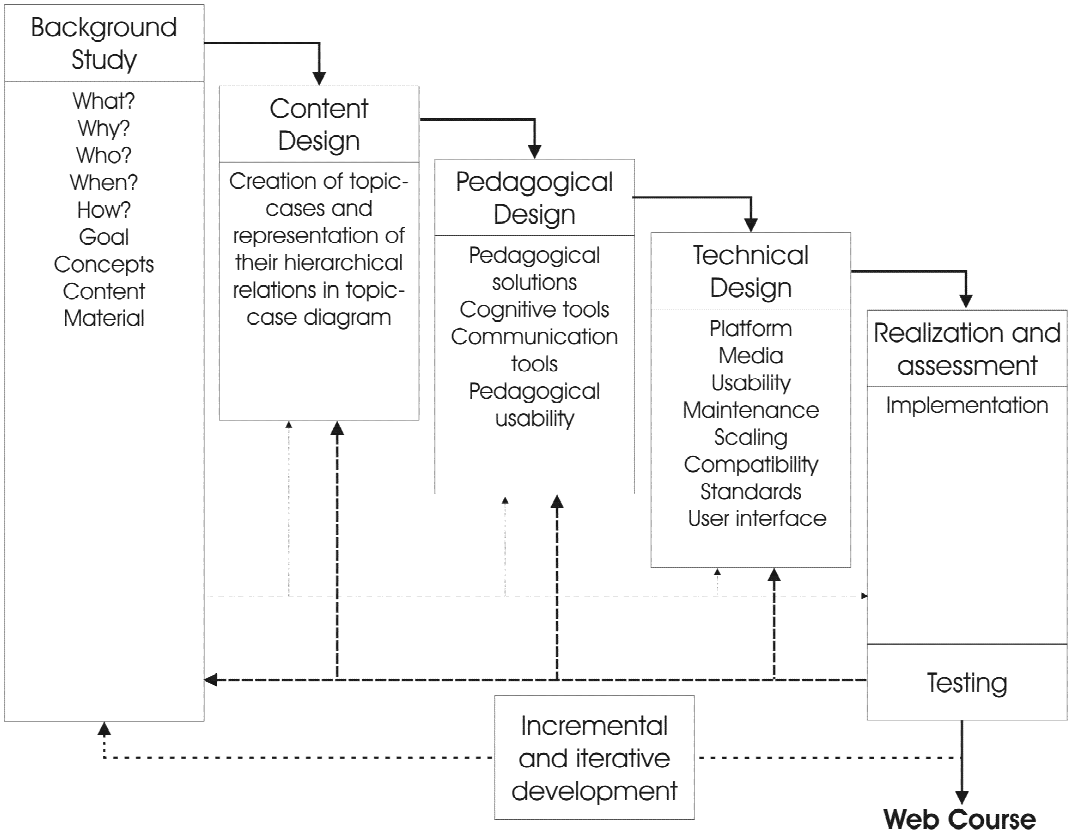


Figure 5. Phases of topic-case driven web-course design and realization process.

4 Background study

The first phase in our web-course design and realization process is the background study. Similarly to White and Montilva, you have to consider several issues before starting to build a web-course.

A central task in the background study is to define and consider all those issues that affect the feasibility of the planned web-course. These kinds of issues are, e.g.

- Why are you designing a web-course? What are the benefits compared to a traditional classroom course? (In our approach a web-course is not a must but an option and/or a possible enhancement of the traditional classroom course.)
- How are you using the web? What is the role of the web? Is the course (or actually parts of it) going to be an output (static) or a process (dynamic)? (Hein et al., 2000) How highly structured the course (actually parts of it) be and is there going to be dialogue or not? (Moore, 1983)
- What is the target group? Who are the students?
- How much time and resources do we have?
- What is the basic idea, focus and goal of the course?
- How do you handle copyrights and agreements, e.g., concerning content creation?

All information about organizational design principles, copyright regulations and design standards should also be considered and documented for further use.

During the background study a useful technique for creating a general view on contents of a course is the concept mapping (Novak, 1998), cf. Figure 6.

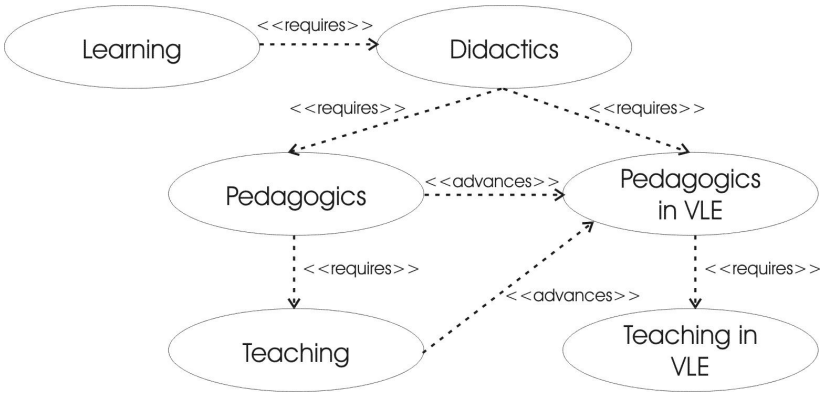


Figure 6. A part of concept map on Virtual Learning Environments.

As a result of this phase you should have a project plan with timetable, resource allocation, financing plan, possible limitations and baselines of the course.

4.1 Software Engineering metaphor: Feasibility study

The background study corresponds to the feasibility study in software engineering processes (e.g., Jaaksi et al., 1999; McConnell, 1998). In software projects, after identifying the scope you should think about whether the project is feasible. Putnam & Myers (1997) points out that “not everything imaginable is feasible.” They also list four solid dimensions of software feasibility: technology, finance, time and resources. These dimensions are also visible in (web) course design. You should find out if your ideas are technically, financially, and *pedagogically* feasible, and whether you have enough time and resources (skills) to implement the whole project.

5 Content design

During the content design phase one designs and documents the basic content of the web-course. This phase is divided into two activities: describing topics of a web-course on a general level and finding relations between individual topics.

5.1 Description of topics

Creation and documentation of topics is obtained through the following basic steps:

1) Generation of the basic set of topics with basic attributes
2) Selection, modification and possible combination of the basic topics to create a non-overlapping structured description of contents

Our method to describe the contents of a web-course at a general level is topic-case. Topic-case is a short but structured description of basic lines of the single course topic (or the course itself in the beginning). With topic-cases we first describe the necessary issues that should be treated during the course. Hence, topic-cases form the skeleton of contents of the course (cf. software architecture). Later we add more features to them, such as pedagogical ideas concerning the realization of topic-cases.

Topic-cases can be documented using suitable forms capturing the necessary attributes during the cumulative development process. The initial topic-case descriptions (see Figure 7) can be formed during the early planning stage (central ones already during the background study). One begins by creating separate topic-cases (using independent and possibly distributed team of content experts if desired) from single issues and then linking them according to preliminary knowledge and pursued learning.

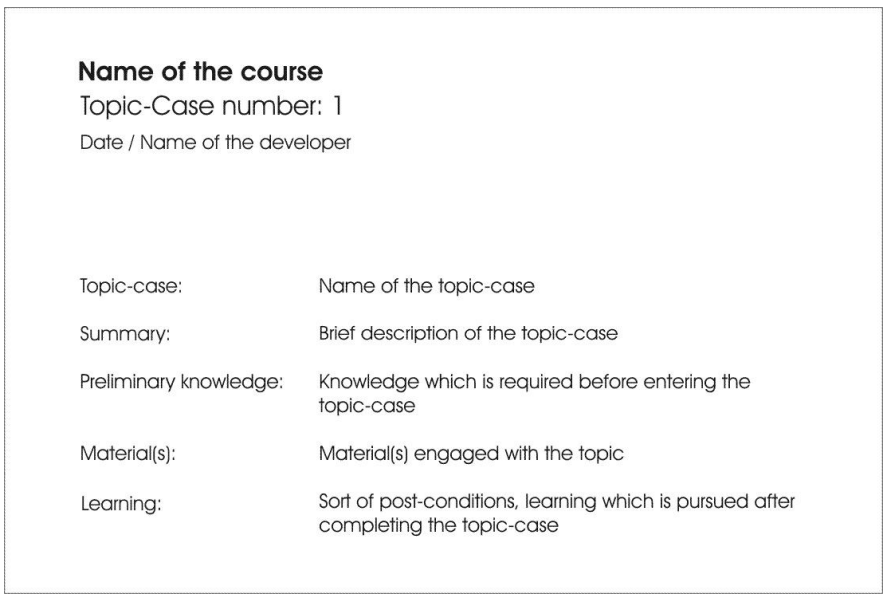


Figure 7. Form of basic description of topic-case.

Topic-cases are authenticated with numbers (and names) that also describe the amount of topic cases, help to evaluate the timetable of the course, and can be used for defining the presentation order of topic-cases. Naturally names and creators of topic-cases should also be documented. Materials engaged with topic-cases can be in any form e.g., books, articles, video clips, recordings. We notice that Humphrey (1998) uses similar kinds of course descriptions in connection with software engineering courses with four attributes: objectives, prerequisites, course structure and course support. Formally, the definition of a set of attributes defines the topic-case interface (cf. component and object interfaces in SE).

5.2 Software Engineering metaphor: Use-case

Jacobson et al. (1999) describes the use-case driven software process where use-cases are used to capture different requirements, bind development processes together, and help to facilitate iterative development. In use cases there are usually "attributes" like actor(s), summary, preconditions, operations and post-conditions. Unlike in use-cases we do not use actors in topic-cases at this point, because their roles depend on the pedagogical solutions to be developed later.

Role of	Unified Process	Topic Case Driven Process
Use-Case / Topic-Case	"a piece of functionality that gives a user a result of value"	a piece of content that gives a student a topic of value
Use-Case Diagram / Topic-Case Diagram	"Describes complete functionality of the system ... What is the system supposed to do (for each user)?"	Describes complete content of the web-course ... What is the web-course supposed to teach (for each student)?

Table 1. Similarities between use-case and topic-case approaches.

Use-cases are the drivers of software development in UP, so we introduce topic-cases as contents development drivers for the web-course design. In Table 1 we present the intimate relations between these two approaches.

5.3 Relations between individual topics

After we are finished with the first set of individual topics we have to find possible relations between them to decide which one should be developed further during the current iteration. The creation of these relations is based on the prerequisite knowledge and pursued learning of each topic-case as documented in the basic form. First, though, one seeks the subgroups of topics that are so similar that it is better to merge and join them together as a single topic-case.

The topic-case relations are represented in the topic-case diagram, which defines the basic contentual hierarchy of the web-course, serving as a more precise content map (see Figure 8). For describing the relations between different topic-cases we introduce new stereotypes: «requires» and «advances». «requires» indicates what knowledge is required before certain topic-case can be accomplished properly. «advances» indicates the knowledge that would be useful to be available, but is not compulsory for the following topic-case.

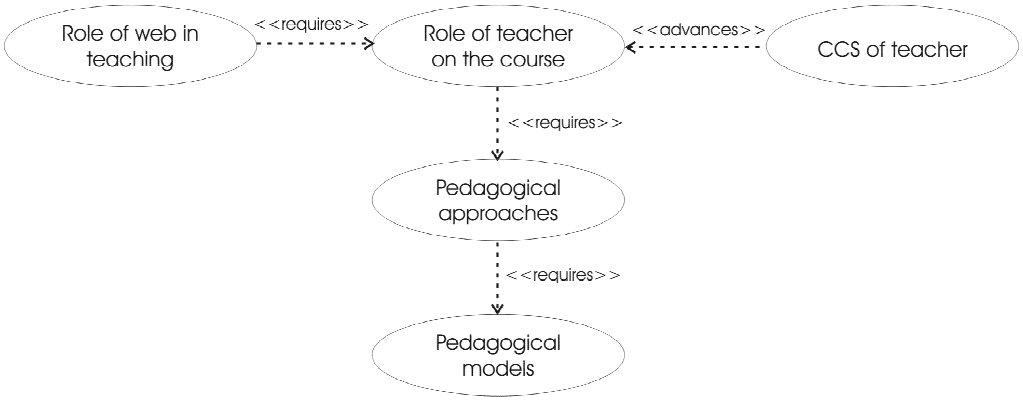


Figure 8. An example of a topic-case diagram with 5 topic-cases.

Using the given stereotypes topic-case diagram reveals which topic-cases are essential to the main concepts in the concept map of the course and which are prerequisites for other topics. This yields a natural way to select those topic-cases that must be implemented first (under the time and resource limitations). The remaining topic-cases that extend the basic knowledge can be implemented during the later iterations or can be used as a subject of term papers or exercises.

This activity should result in a topic-case diagram that describes realizable topics and relations between them. After the content design phase all topic-case descriptions, with a topic-case diagram, present the basic content of the web-course.

5.4 Software Engineering metaphor: Use-cases and use-case diagram

Jacobson et al (1999) defines the use-case diagram as a model that "describes the complete functionality of the system." This model answers the question: What is the system supposed to do (for each user)? An example of a use case diagram is shown in Figure 9.

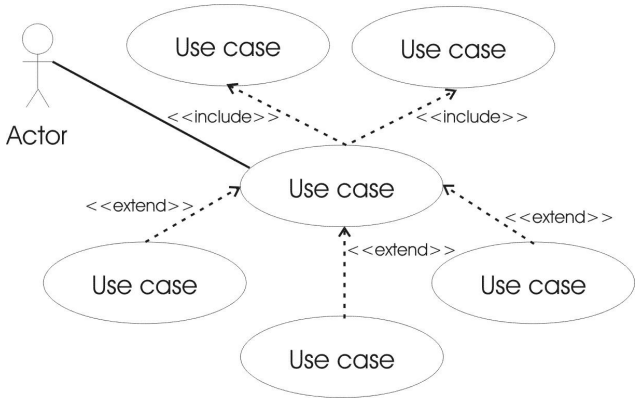


Figure 9. An example of a use-case diagram with one actor and six use cases.

Use-case diagrams usually contain three kinds of stereotypes: «uses», «include» and «extend». These notations describe the nature of relation between two use-cases.

Before creating the use-case diagram one needs to do some kind of evaluation and select limited amount of realizable use-cases. Bass et al. (1998; Chapter 9) present an evaluation process of scenarios (a scenario is a brief description of some anticipated or desired use of the system) in connection with choosing an appropriate software architecture. Their approach is readily applicable for evaluation of use-cases (as well as topic-cases). Evaluation process of Bass et al. proceed as followed:

- *Classification of scenarios:* can the system execute scenarios directly or indirectly (without any modifications to the system or some modifications are required)?
- *Individual evaluation on scenarios:* what kinds of changes are needed for the system to support the scenario?
- *Assessment of scenario interaction:* what kind of interaction is required between scenarios and the system?
- *Overall evaluation:* how important the scenario interactions are for the activities you expect the system to be able to perform?

6 Pedagogical design

Pedagogical design is usually forgotten in web-course design. One reason for this could be the fact that pedagogical design is definitely not easy. Another reason could be that usually web-designers do not have a pedagogical background. In our process the individual phase for pedagogical design ensures that this issue, which should underlie all teaching activities, has its special role within web-course design. The preliminary knowledge behind a successful pedagogical design was summarized in Section 2.

6.1 Questions behind pedagogical design

What kind of learning do we support in our web-course: instrumental, communicative or emancipatory? What is a suitable approach for each topic: instructional, cognitive, constructive, humanistic or critical humanistic learning? What forms of activity do we use? What kinds of media do we utilize in our course? These are the basic questions which have to be considered next.

Pedagogical problems are also related to the roles of teachers and students, ways of teaching and learning and actions in those situations, learning tasks with different characteristics, guidance and control in learning, assessment, and feedback. In addition, pedagogical design includes the integration of communication and cognitive tools into the web-course and consideration of pedagogical usability. All of these pedagogical issues should be considered and written down during the pedagogical design phase. Some of these issues have been brought up already during the background study, but at this point each topic is connect to its pedagogical solution.

6.2 Pedagogical solutions for each topic

After the content design phase we have selected topics for the current iteration that need some kind of pedagogical activities to support and to describe teaching and learning of that topic. In this phase we augment these topics with advisable pedagogical activities.

To document the decisions made we use extended topic-case descriptions (see Figure 10) where we add a few more attributes that describe in more detail what teachers and students should do and what kind of teaching and learning activities are recommended. New attributes: actor(s), description, pedagogical solution(s) and relations, are filled in each topic-case.

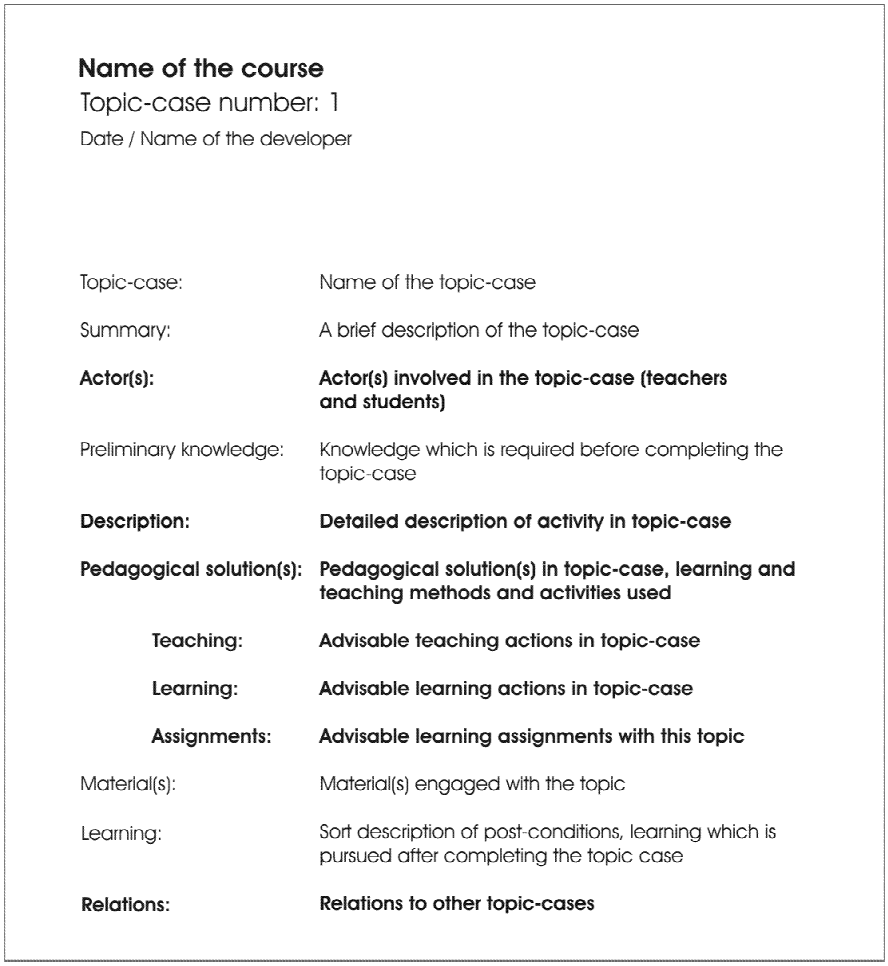


Figure 10. Form of the extended topic-case.

Each topic-case may have more than one possible pedagogical solution. In some cases it might be better for (different) students to be able to see a topic from different perspectives. Pedagogical solutions contain both teaching and learning activities, and recommended assignments for learning session.

Relations to other topic-cases are important. These relations identify links between different topics for the technical design. They are also useful if we have to update the contents later on.

After this phase you should have documented the content of the web-course and pedagogical activities for each topic, which are represented in the extended topic-case descriptions.

6.3 Software Engineering metaphor: Usability design

In software projects usability is one of those issues that should be considered during the process in order to develop a usable and well-designed product. Hakiel (1997) presents two contrasting approaches regarding how usability can be integrated with software engineering:

1) Usability design deliverables are aligned with software design deliverables
2) Usability design deliverables are contributing to software requirements

These approaches conform also to pedagogical design; pedagogical issues should and can be taken into account at the beginning of the process or during the pedagogical design. If a web-course design is based on strong pedagogical models, these can be the ultimate drivers for the whole design process (see Figure 11). Concerning the extended topic-cases in Figure 10, this just means that the necessary attributes are filled out in a different order.

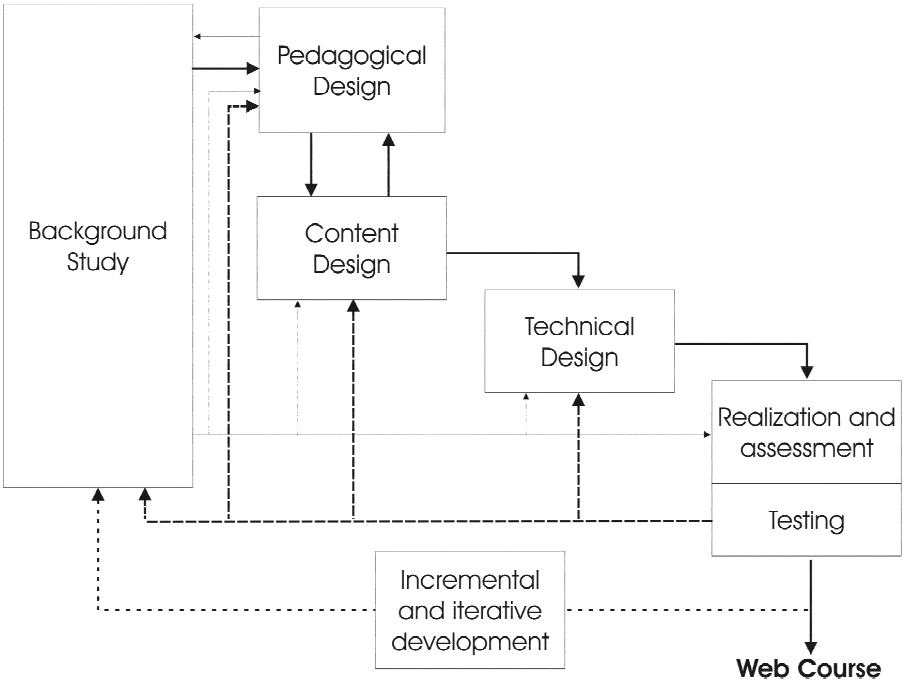


Figure 11. Form of design process when strong pedagogical models are driving the whole topic-case driven design process.

7 Detailed technical design

We separate the technical design from the pedagogical design, because the pedagogical issues definitely deserve special attention of their own. In the technical design phase we should make decisions concerning technical issues, like use of platform, media, maintenance, scaling, compatibility, user interface, etc. During this phase, we should also keep usability issues (see Chapter 2.6) and different standards, e.g., LOM (2002) and ISO 13407 (see Chapter 2.7) in mind.

We do not want to commit ourselves here to certain technical decisions, because there are again many ways to implement a web-course. It is possible to implement the web-course based on, e.g., databases, simple HTML web pages, XML or a combination of all of these. There might also be limited tools and software available in different projects. So, how the web-course is actually implemented is based on resources and knowledge in use. In the next sections we say a few words about different tasks concerning technical issues.

7.1 Use of platform

Web-courses can be implemented as open web pages or using some specific platform. Nowadays there are plenty of different kinds of platforms in use all around the world. The most commonly used platforms are perhaps WebCT (see http://www.webct.com), TopClass (see http://www.wbtsystems.com) and IBM Lotus Learning Space (see http://www.lotus.com). All platforms have different features and different basic rules. They all enable material delivery and communication between individuals, but some of them even support and promote learning.

More precisely, there are usually some tools that support cognitive processes and communication. Platforms usually also offer certain ways to transmit learning materials, but too often only in the platform's own format. Comparison of existing online course delivery software products is difficult, but many reported comparisons have been made and are available on the Web (see, e.g., http://www.edutools.info/landonline/).

At the University of Jyväskylä we now use a Finnish platform product called Discendum Optima (see http://www.discendum.com/english/index.html), which supports web-course design through:

- A simple and explicit interface
- Ease of use for both students and teachers
- Easy maintenance
- Reusable objects
- Use of external resources (e.g., HTML, Word, pdf)
- Transferable objects in HTML format

7.2 Medias in use

Depending on the kinds of contents and topics in our web-course it is possible to use different kinds of media for presentation. The typical medium is written text, but many times it would be more illustrative to use photos, graphs, tables, animation, videos or even simulations to present current information. These other media are often more informative, but can be much more expensive to produce. They also have higher technical requirements that might be too hard to reach without an expert in the field. Notice that there are also guidelines for writing for the Web, e.g., by Morkes and Nielsen (1997).

7.3 Maintenance, scaling and compatibility

In technical design there are three essential issues that should be considered in order to create web-courses with lasting life cycles. These issues are maintenance, scaling and compatibility. Maintenance includes, e.g.:

- Updating date-sensitive materials such as timetables and schedules
- Modernizing the outlook
- Keeping contact information current
- Adding new information or features
- Updating user information

A Web-course should be easy to maintain: files that need to be updated often or continually could be in the same folder, files should be organized with some systematic regulation, files and web pages should be named in a recognizable way, etc. All of these issues rely on well-structured contentual organization of the material.

Scaling means that the learning system is capable of presenting multiple courses with hundreds of topic-cases for thousands or even tens of thousands of students concurrently and simultaneously. Solutions for this purpose are out of the scope of the present work, but basically they are always based on necessary improvements on hardware or software.

The contents of the web-course should also feature compatibility with different platforms and other systems in use. In many occasions one needs to convert parts of the web-course or even the whole course into the new environment (or platform). For this reason it would be better to avoid special, platform-dependent formats in the material production.

7.4 User interface

Jacob Nielsen has written many books about web usability (e.g., Nielsen, 2000). His advice is also useful in web-course design. The user interface is the most immediately visible part of web-course and users are usually looking at a single page at a time. Especially, when we are designing a web-course, “web pages should be dominated by content” not by outfit (Nielsen, 2000). According to Nielsen (2000) “navigation is a necessary evil that is not a goal in itself and should be minimized.” However, users should always know where they

are, where they are coming from and where they can go. It is also important for users to know if they have definitely logged out from the system where they logged into with personal user identification. Some tips provided by Nielsen for interface design were reviewed in Section 2.8.

7.5 Software Engineering metaphor: Design

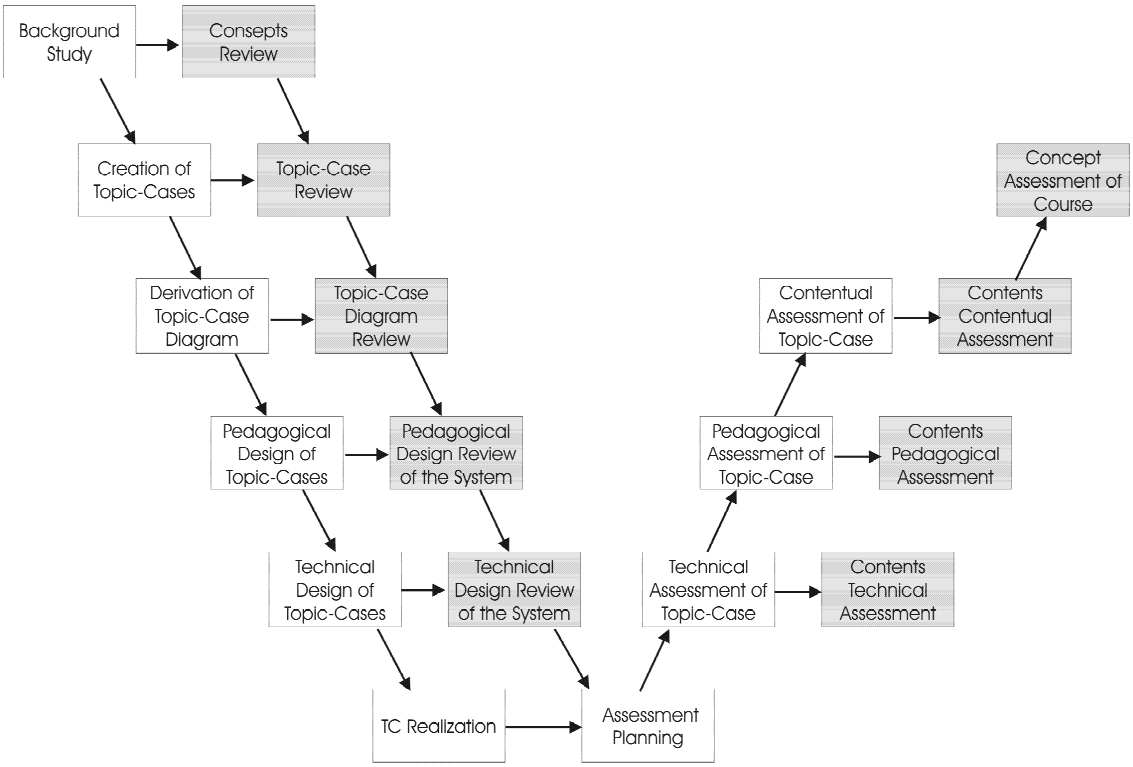
During the design phase in software projects the system finds its final structure that guarantees the fulfillment of all requirements. In the Unified Process this phase yields a description regarding how each use case for the current iteration is going to be realized and what kind of an interface the user has for the desired functionality.

8 Realization and assessment

The final realization or implementation consists of completing the individual topic cases using the chosen pedagogical and technical solutions. This means that the contents is enlarged to the final length and teaching and learning actions are described in details in connection with the final contents and the media in use.

Assessment is an activity that should be an essential part of the whole development process (and the maintenance phase as well). We divide the overall assessment (see Figure 12) into three parts:

- Reviews during the development phase
- Assessment of topics and contents after realization
- Assessment of user's required technical, pedagogical and contentual skills



Picture 12. Assessment of the web-course in accordance with modified V-model of testing.

8.1 Reviews

Reviews are carried out at the end of each step by developers to ensure that everything has been done as required and to locate as many errors and open problems as possible before proceeding to the next step. For instance, derivation of the topic-case diagram is reviewed

by comparing it to the output of the prior step (the creation of topic-cases), and feeding back any discovered mistakes.

8.2 Planning and performing of assessment

An actual assessment plan is made after the realization of topics. The assessment plan should cover both the assessment of single topics and the assessment of the whole content of a web-course. It should also assess the user's required technical, pedagogical and contentual skills. Assessment is performed by users (students, teachers and technical staff) with real learning, teaching and maintaining assignments.

Assessment of single topics and assessment of the whole content are both divided into three steps: technical assessment, pedagogical assessment and contentual assessment.

8.2.1 Technical assessment

In the technical assessment, technical realization of a single topic is assessed first and then technical functionality of the whole system is considered. Technical assessment is based on five questions (see Nielsen's usability attributes in Chapter 2.8.):

- Is the use of the web-course easy to learn?
- Is the web-course efficient to use?
- Is the use of the web-course easy to remember?
- Does the web-course have a low error rate and is it easy to recover from those errors?
- Is the web-course pleasant to use?

8.2.2 Pedagogical assessment

Pedagogical assessment is based on meaningfulness of learning and pedagogical usability (see Chapter 2.8.) including e.g. the following questions:

- Does the web-course support mindful thinking and knowledge presentation?
- Does the web-course support communication with others?
- Does the web-course offer accessibility to information and construction of personal representations?
- Does the web-course support social negotiation and forming communities of learners?
- Does the web-course offer a proper articulation of goals, willful achievements and mindful effort?
- Does the web-course support the forming of knowledge building communities?
- Does the web-course support the solving of real-world tasks, meaningful and complex problems, constructing situation-specific schemas and defining/interacting with problem space?
- Does the web-course support articulation and reflection of new knowledge?

In our approach, pedagogical assessment includes also checking that all topic-cases have a pedagogical solution that is consistent with the underlying pedagogical models for the particular web-course.

8.2.3 Contentual assessment

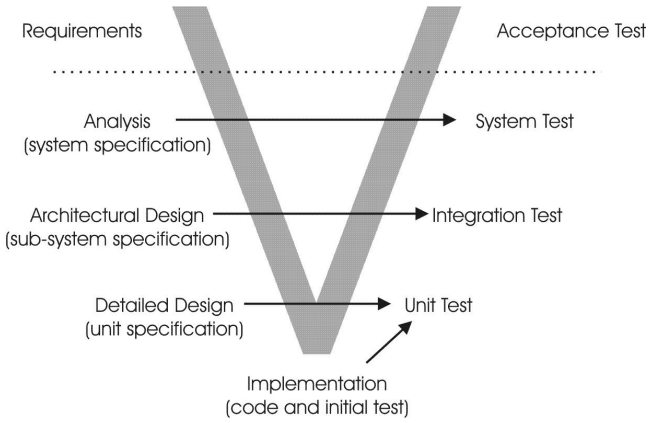
The main issues for the contentual assessment are:

- Does the web-course include all the topics that were planned at the beginning?
- Are all topics linked properly (logically) together?

In our approach this means that all of the selected topic-cases have been implemented properly and relations between topic-cases in the topic-case diagram are consistent with the corresponding plans.

8.3 Software Engineering metaphor: Testing

In software engineering projects there are many different ways to carry out the software testing. One of the most used testing methods is the V-model which was first introduced in 1979 by Glenford Myers (1979). This model is presented now in a slightly different way, but the basic idea is the same (see Figure 12). In the V-model, "the testing cycle has been structured to model the development cycle" (Myers, 1979). The key ingredient of this metaphor is the relation of the development of different conceptual phases with their test goals.



Picture 13. V-model of software testing.

9 Creating a web-course repository

Usually there are a lot of brilliant topics to affiliate on the web-course during the first phase of the web-course development process but time and resources are limited. Therefore, you have to make choices between topics and prioritize which topics, forming the core of the web-course, are to be implemented first. You can start your web-course design with small amounts of the most important topics in the first iteration and add more topics during following iterations. All of this requires good planning, documentation and some kind of standardized procedure to work out, which is precisely supported by the topic-case driven approach presented in this paper.

9.1 Reuse of learning objects

LOM (2002) defines learning objects, or small instructional components as, “any entity – digital or non-digital – that may be used for learning, education, or training.” The reuse of learning objects means that the same learning object can be used in multiple contexts for multiple purposes.

In our approach, all phase deliverables (basic topic-cases, topic-case diagram, extended topic-cases etc.) can be defined as learning objects. More precisely, topic-cases and topic-case diagrams (or parts of them) can be reused in some other context or in some other web-course. In different contexts one can apply different pedagogical and technical solutions, which can be easily changed and/or added into extended topic-case descriptions.

Topic-cases can also be extended, with new attributes if needed, in the identification of reusable learning objects or to support some other standards and technical constraints.

9.2 Extension of content

The topic-case driven development process allows for the iterative and incremental development of web-courses. Once you have mastered all of these phases for the first time and built a structure with relations for smaller contents, you can add new topics into a web-course during the next iteration.

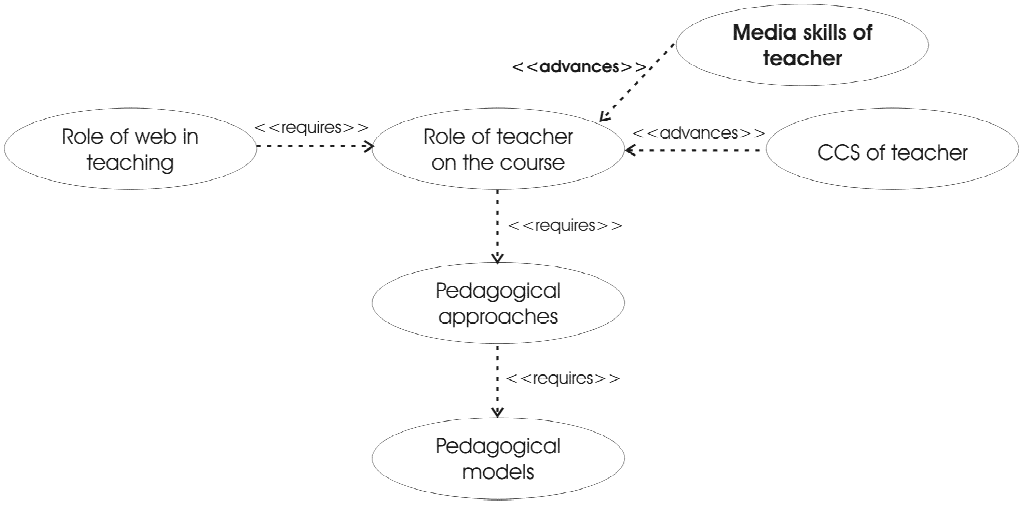


Figure 14. Extension of content with one new topic-case (see also Figure 8).

First you make a new project plan in the background study phase (time and resources have to be considered). Then you make new topic-case descriptions from new topics and add them into the topic-case diagram (see Figure 14) in the content design phase and/or choose to realize those topics from the existing diagram that were left out from the previous iteration. The chosen topic-cases are then augmented with pedagogical solutions during the pedagogical design phase and fitted into the technical design during the next phase. At the end you evaluate the whole web-course again.

9.3 Towards a web-course repository

The topic-case driven development process is an iterative and incremental work flow that naturally supports the creation of a web-course repository, where the existing topics are related to different courses, through which the overall topic-case diagram (and/or through the overall content map) are contentually related to each other.

After you have designed and implemented several web-courses you have, not only a lot of reusable topic-cases, but also several solutions for pedagogical and technical issues. From these different objects and solutions you can create a web-course repository with reusable learning objects and pedagogical and technical solutions for next web-courses.

In the event that you have difficulties in solving pedagogical or technical issues on web-course design, you can explore this web-course repository and find solutions or at least develop some new ideas. You can also integrate old ideas into new web-courses. Management of resources is also easier with a repository. A Web-course repository is also extensible: you can add new topic-cases, pedagogical and technical issues into repository at any time.

To this end, the topic-case driven approach can be applied, in addition to the web-course and repository design, to the training program design. Topic-cases can be used to define an individual course and content maps/general topic-case diagrams can be used to define relations between different courses. As a result you are able to produce a pedagogically designed and assessed training program.

References

Bass, L., Clements, P. & Kazman, R. (1998). Software Architecture in Practice. The SEI Series in Software Engineering. MA: Addison-Wesley.

Boehm, B., Egyed, A., Port, D., Shah, A., Kwan, J. & Madachy, R. (1998). A shakeholder win-win approach to software engineering education. Annals of Software Engineering, Vol. 6, pp. 295-321. Kluwer Academic Publishers. Available on the Web: http://manta.cs.vt.edu/ase/

Catenazzi, N. & Sommaruga, L. (2002). Re-using contents a recipe for e-learning. In Proceedings of 4th ICNEE, 8th - 11th May 2002, Lugano, Switzerland, Session "Pedagogical and Social Issues" 2.1/11.

Council of European Union (2000), eContent Programme. Available on the Web: http://www.cordis.lu/econtent/

Finnish Ministry of Education (2002). Digital Content Creation 2007 - Strategic aims and action. Available on the Web: http://www.minedu.fi/opm/hankkeet/sisu/DigitalContentCreationStrategy1.doc (In Finnish)

Hakiel, S. (1997). Usability engineering and software engineering: How do they relate? In Smith M.J., Slavendy G. and Koubek R.J. (eds.). Advances in Human Factors/Erconomics, 21B Design of Computing Systems: Social and Ergonomic Considerations. Proceedings of Seventh International Conference on Human-Computer Interaction, San Francisco, California, USA. Vol. 2. Elsevier, pp. 521-524.

Hein, I., Ihanainen, P. & Nieminen, J. (2000). Tunne verkko. In OTE - opetus & teknologia, 1/2000, s. 5-8. Opetushallitus. (In Finnish)

Hoover, C.L. (1998). TAP-D: A model for developing specialization tracks in a graduate software engineering curriculum. Annals of Software Engineering, Vol. 6, pp. 253-279. Kluwer Academic Publishers. Available on the Web: http://manta.cs.vt.edu/ase/

Horila, M., Nokelainen, P., Syvänen, A. & Överlund, J. (2002). Pedagogisen käytettävyyden kriteerit ja kokemuksia OPIT-oppimisympäristön käytöstä Hämeenlinnan normaalikoulussa syksyllä 2001. DL-projektin osaraportti. Hämeen ammattikorkeakoulu, Hämeenlinna. Available on the Web: http://www.hamk.fi/julkaisut/julkaisu.php?id=287 (In Finnish)

Humphrey, W.S. (1998). Why don't they practice what we preach? Annals of Software Engineering, Vol. 6, pp. 201-222. Kluwer Academic Publishers. Available on the Web: http://manta.cs.vt.edu/ase/

Häkkinen, P. (1996). Design, Take into Use and Effects of Computer-Based Learning Environments - Designer's, Teacher's and student's Interpretation. Academic dissertation. University of Joensuu, Publications in Education, N:o 34.

ISO 13407: 1999. Human-centered design process for interactive systems.

ISO 9241-11: 1998. Guidance on Usability.

Jaaksi, A., Aalto, J-M., Aalto, A. & Vättö, K. (1999). Tried & True Object Development - Practical Approaches with UML. Cambridge University Press.

Jacobsen, P. (2001). Reusable learning objects - What does the future hold? E-learning Magazine, Nov 1, 2001. Available on the Web:

http://www.elearningmag.com/elearning/article/articleDetail.jsp?id=5043

Jacobson, I., Christerson, M., Jonsson, P. & Övergaard, G. (1992). Object-Oriented Software Engineering: A Use-Case Driven Approach. MA: Addison-Wesley.

Jacobson, I., Booch, G. & Rumbaugh, J. (1999). The Unified Software Development Process. Addison-Wesley.

John, B.E., Miller, P.L., Myers, B.A., Neuwirth, C.M. & Shafer, S.A. (eds.) (1992). Human-Computer Interaction in the School of Computer-Science. Technical Report CMU-CS-92-193, Carnegie Mellon University, Pittsburgh, PA.

Jonassen, D.H. (1992). What are Cognitive Tools. In Kommers P.A.M., Jonassen D.H. and Mayers J.T. (eds), Cognitive Tools for Learning, Berlin: Springer Verlag, pp. 1-6.

Jonassen, D.H. (1995). Supporting communities of learners with technology: a vision for integrating technology with learning in schools. Educational Technology, July-August 1995, pp. 60-63.

Lajolie, S.P. (1993). Computer Environments as Cocnitive Tools for Enhancing Learning. In Lajolie, S.P. and Derry, S.J. (eds), Computers as Cocnitive Tools, Lawrence Erlbaum Associates, Publisher, pp. 261-288.

LOM (2002). IEEE Standard for Learning Object Metadata, IEEE Std 1484.12.1-2002.

Manninen, J. & Pesonen, S. (2001) Aikuisdidaktiset lähestymistavat, Verkkopohjaisten oppimisympäristöjen sunnittelun taustaa. In Matikainen J. and Manninen J. (eds.) Aikuiskoulutus verkossa, Verkkopohjaisten oppimisympäristöjen teoriaa ja käytäntöä Palmenia-kustannus, pp. 63-79. (In Finnish)

McConnell, S. (1996). Rapid Development. Microsoft Press, Redmond, Wa.

McConnell, S. (1998). Feasibility Studies. IEEE Software, Vol. 15, Number 3, pp. 119-120.

Mezirow, J. (1981). A Critical Theory of Adult Learning and Education. Adult Education, Vol. 32, Number 1, Fall 1981, pp. 3-24.

Montilva, C. J.A. (2000) Development of Web-based courses: A software engineering approach. Presented in Symposium on 21st Century Teaching Technologies: A Continuing Series of Explorations, March 24, 2000. Available on the Web:

http://www.fmhi.usf.edu/usfsymposium/ 2000/handout/montilva.pdf

Moore, M.(1983). On a Theory of independent study. In Seward D., Keegan D., and Holmberg B., (eds.) Distance Education: International Perspectives London: Croom Helm, pp. 68-94.

Morkes, J. & Nielsen, J. (1997). Concise, SCANNABLE, and Objective: How to Write for the Web. Available on the Web: http://www.useit.com/papers/webwriting/writing.html

Multisilta, J. (1997). Miltä näyttää WWW-maailma oppimisympäristönä. In Lehtinen E. (ed.) Verkkopedagogiikka. Edita. (In Finnish)

Myers, G. (1979). The Art of Software Testing. A Wiley-Interscience Publications.

Nielsen, J. (1993). Usability Engineering. Academic Press, London.

Nielsen, J. (2000). Designing Web Usability. New Riders Publishing, Indianapolis, USA.

Novak, J.D. (1998). Learning, creating, and using knowledge: concept maps as facilitative tools in schools and corporations. Lawrence Erlbaum Associates.

Pekkola, S. (2003). Multiple Media in Group Work - Emphasising Individual Users in Distributed and Real-Time CSCW Systems. Jyväskylä Studies in Computing, Number 29, University of Jyväskylä. (PhD-Thesis)

Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland S. & Carey T. (1994). Human-Computer Interaction. Addison-Wesley, Harlow.

Putnam, L. & Myers, W. (1997). How Solved is the Cost Estimation Problem. IEEE Software, November 1997, pp. 105-107.

Smolander, K., Tahvanainen, V.-P. & Lyytinen, K. (1990). How to Combine Tools and Methods in Practice - a field study. In Proceedings of Advanced Information Systems Engineering, Second Nordic Conference CAiSE'90, Stockholm, Sweden, May 8-10, 1990, pp. 195-214.

Tella, S., Vahtivuori, S., Vuorento, A., Wager, P. & Oksanen, U. (2001). Verkko opetuksessa - opettaja verkossa. Edita. (In Finnish)

White, S.A. (2000). Experience with a process for software engineering web-course development. In Proceedings of 30th ASEE/IEEE Frontiers in Education Conference, T1C, pp.13-18.