Fraunhofer Institut
Experimentelles
Software Engineering

Improving Courseware Quality through Life-Cycle Encompassing Quality Assurance

Authors:
Ines Grützner
Patrick Waterson
Steaphan Weibelzahl

Submitted for Publication at
ACM SAC `04 Track on Engineering
e-Learning Systems, Nicosia,
March 14-17, 2004

IESE-Report No. 081.03/E
Version 1.0
December 2003

A publication by Fraunhofer IESE

Fraunhofer IESE is an institute of the Fraunhofer Gesellschaft.
The institute transfers innovative software development techniques, methods and tools into industrial practice, assists companies in building software competencies customized to their needs, and helps them to establish a competitive market position.

Fraunhofer IESE is directed by
Prof. Dr. Dieter Rombach
Sauerwiesen 6
67661 Kaiserslautern

Abstract

The quality of courseware development is affected by four factors: content and instructional issues; management; technical and graphical issues; and concerns of the customer. In this paper we describe IntView, a courseware development method that integrates these four factors throughout the whole development life-cycle. By combining existing courseware quality assurance methodologies with software engineering techniques such as inspections and tests the interests of the participating roles are balanced. Both the IntView methodology and the quality assurance techniques are described and the results of some preliminary case studies are reported.

Table of Contents

1 Introduction

Technology-based learning, for example via the Internet, is becoming more and more important particularly within the world of business where access to learning materials needs to be brought about quickly and efficiently. Despite these developments one of the main barriers to successful deployment of technology-based learning is the poor availability within the marketplace of high quality courseware that is tailored to the needs of individual users and groups [14]. Courseware in this context includes all kinds of educational material and content that is distributed via the web for training purposes from the users' point of view, as well as collections of multimedia documents interrelated by means of navigational structures. Its quality is chiefly concerned with four main dimensions, these are: the content of learning materials; the presentation of these materials; the way in which they are taught (i.e., *pedagogic* content); and, the overall functionality of the courseware. All four dimensions have to be considered at the same time continuously throughout the life-cycle of the courseware to ensure that the final product is of high quality and thus to facilitate a high learning gain.

However, current courseware development approaches do neither support continuous quality assurance of all development artifacts and the courseware to be produced itself nor do they assure the quality in all four courseware dimensions equally. Many approaches propose only evaluations of the fully implemented courseware (e.g., [5]), or tests (e.g., [18]). The major drawback of this "late" quality assurance is that solving problems introduced in the requirements specification phase or in the design phase is much more expensive than solving them in the phases they were introduced [2]. Other quality assurance approaches that have been proposed include prototyping [3] or goal detailing [5]. These measures assure the quality of the courseware in early development stages. However, they cover only a few artifacts. To sum up, what is still missing in courseware engineering is a life-cycle encompassing quality assurance methodology that covers all development artifacts and the final courseware as it is proposed in software engineering.

This paper presents such a life-cycle encompassing quality assurance methodology that is an integral part of the courseware engineering methodology (IntView). This methodology is a unique combination of already existing courseware quality assurance methodologies like prototyping and evaluation with software engineering quality assurance methods like inspections and tests, which were extended and adapted to cover all four courseware dimensions. In the second part of the paper, we give a short summary of the phases, roles and products that are involved in this methodology. Section 3 provides a detailed

description of the four elements of the quality assurance methodology. Finally, in section 4, first results from the application of the quality assurance methodology in courseware development projects are presented.

2 Overview of the IntView courseware engineering methodology

IntView is a courseware engineering methodology, i.e., it supports a systematic courseware development process. The IntView methodology is designed in order to support and integrate all of the different perspectives and views that are involved in the process of designing high quality courseware. These views encompass management, content development, pedagogic, as well as technical inputs into the process as a whole. The way in which this is achieved draws upon existing work in the literature on the development of learning software (e.g., [11], [13], [15]), software engineering (e.g., [4]), and hypermedia engineering (e.g., [6], [12]). Much of the content of the quality assurance component of the methodology is adapted from previous work within the domain of software engineering, for example from inspections [10]. Figure 1 presents a high level overview of the methodology in the form of a life-cycle model (based upon the overall development process for web applications defined by Ginige [6]).

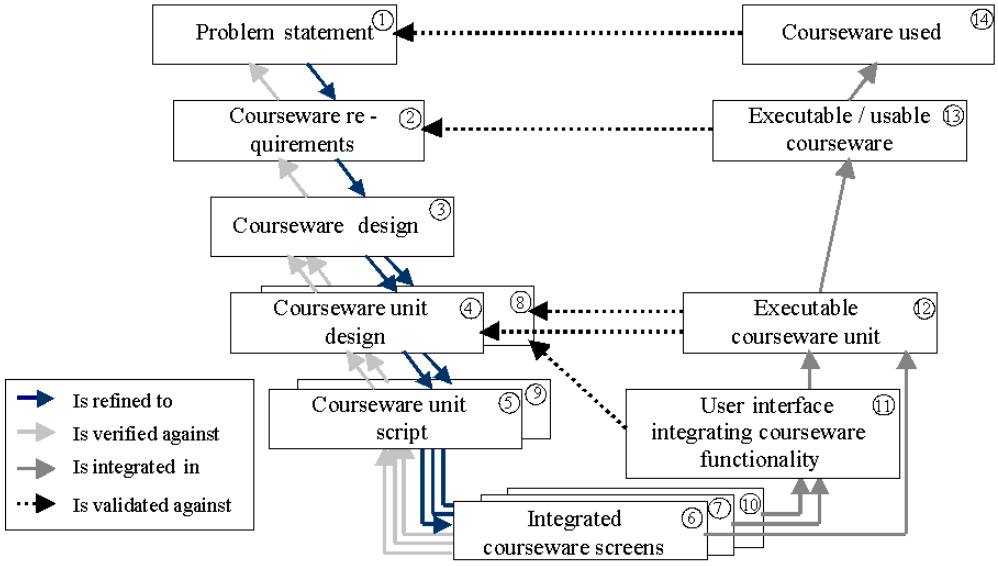

Figure 1: The product-centered IntView life-cycle model

Table 1: Overview of the refinement phases of the IntView methodology. The input artifacts that are required for a certain phase, the activities performed during this phase and the output artifacts as well as the participating roles are listed. Optional phases are indicated by *. The roles are abbreviated as shown in Table 3.

	Name of the Phase	Input Artifacts	Main Activities	Output Artifacts	Roles
1	Problem specification	Customer or in-house request	Analysis of the planned courseware and its environment Market and technological analysis Preliminary budget and schedule planning Decision on development	Problem statement	PM, C, L, EC, ID, SE, CP
2	Courseware requirements specification	Problem statement	Target group and needs analysis Instructional specification Requirements and courseware architecture specification (including prototyping) Project and evaluation planning Acceptance and system test case specification	Courseware requirements specification	PM, ID, CP, all roles
3	Courseware design	Courseware requirements specification	Content design Instructional design Functional design Media and user interface design Prototyping	Courseware design	CP, HF, ID, SE, all roles
4	Courseware unit design	Courseware design	Learning step specification including media and interaction sketches Courseware unit test case specification	Courseware unit design	SE, ID
5	Courseware unit scripting	Courseware unit design	Courseware unit authoring Media and interaction specification Definition of links, glossary entries etc.	Courseware unit script	CA, SE, ID
6	Media and screen production	Courseware unit script Courseware user interface	Media production Screen production Integration of media into screens	Integrated courseware screens	CP, MD, GD
7	Courseware user interface implementation	Courseware design	User interface implementation Screen and interaction template implementation	Courseware user interface	CP, GD, MD
8*	Courseware functionality requirements specification	Courseware design	Courseware functionality requirements specification (including prototyping) Courseware functionality test case specification	Courseware functionality requirements	P

9*	Courseware functionality design	Courseware functionality requirements	Courseware functionality design Courseware functionality architecture specification	Courseware functionality design	P
10*	Courseware functionality implementation	Courseware functionality design	Courseware functionality implementation Courseware functionality testing	Courseware functionality	P
11*	Courseware functionality integration into courseware user interface	Courseware user interface Courseware functionality	Integration of courseware functionality into courseware user interface Courseware user interface testing	User interface integrating courseware functionality components	CP, P
12	Courseware screen integration into courseware user interface with functionality	User interface integrating courseware functionality components Integrated courseware screens	Courseware screen concatenation Courseware unit testing	Executable courseware unit	CP
13	Courseware unit integration	Executable courseware unit	Courseware unit concatenation System and acceptance test Formative evaluation	Executable / usable courseware	CP, C, L, QM
14	Courseware use	Executable / usable courseware	Summative evaluation Continuous improvement of the courseware	Courseware used	PM; all roles

The life-cycle model defines all of the phases that are necessary to produce all development artifacts required for the development of high quality learning software [7]. Table 1 provides an overview of the activities, artifacts and roles (Table 3) that are involved in each phase.

A detailed presentation of the IntView methodology including descriptions of all activities, products, roles, and tools as well with all product templates will be given in the IntView handbook, which is currently in production. This handbook will be published as printed book as well as electronic process handbook (in the form of an electronic process guide [1]).

3 Quality Assurance with IntView

The courseware engineering life-cycle model of the IntView methodology is accompanied by four engineering methods that assure the quality of the output artifacts of each phase. These methods include perspective based inspections, prototyping, tests, and both formative and summative evaluations. Thus, the whole life-cycle is encompassed by verification and validation tasks that are assigned to different roles.

3.1 Perspective based inspections

The output artifacts of the phases 2-10 are verified against the output of the previous phases by means of perspective based inspections. Each participating role checks whether the artifact (e.g., requirements specification, courseware design, etc.) meets the definitions or specification of the previous phase. Moreover, each role evaluates the product in terms of the previously defined quality criteria.

This concept is illustrated by an example from the target group description in phase 2. The description of the target group and the qualification need analysis are verified against the problem description in phase 1. The following roles participate in the verification process, since they were involved in the creation of the problem description (see Figure 2): The customer, a potential learner and/or an expert of the customer, the instructional courseware designer, the subject matter expert, and the courseware programmer. Moreover, the quality assurance manager organizes and conducts the verification process and the project manager responsible for controlling gives an overview of the participating roles, the input products, the verified (modified) products, and the used quality assurance method.

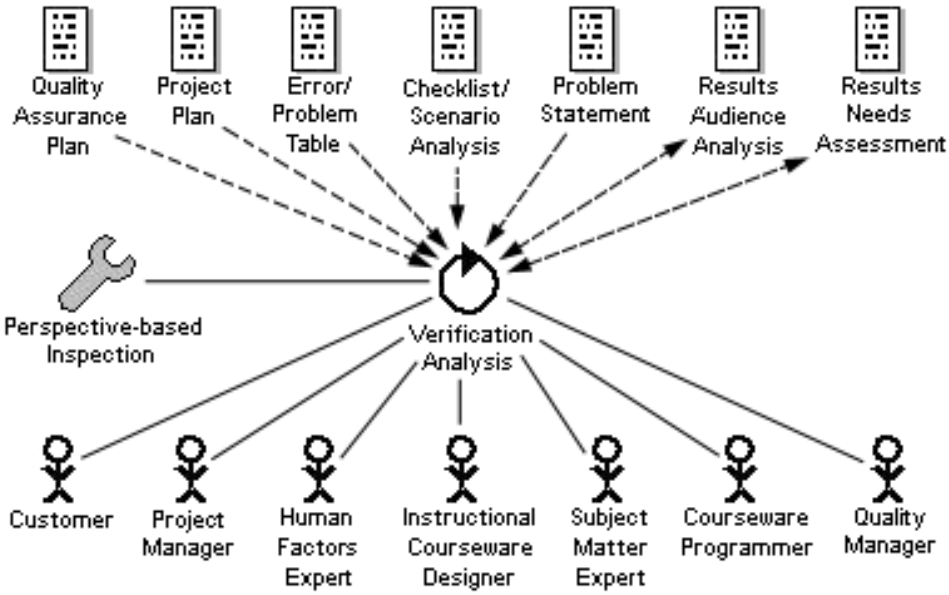

Figure 2: View of the verification process in phase 2, based on the process-modeling schema SPEARMINT

The perspective-based inspections are performed by means of general checklists as well as scenarios that are tailored to the particular product and role. The general checklist contains questions that are important for all roles. For example one of these overall criteria is consistency. Thus, the checklist question in the above example is derived in the following way: "Are the target groups and the qualification need described in a consistent way? "

The tailored scenarios consist of two parts: first, a set of instructions for activities to be performed while reading the product. For example the subject matter expert receives the following instruction: "Please derive the qualification topics from the description of the target group and the needs analysis by using the IntView methodology". Second, the scenario contains several questions that are specific to the product and the qualification process. In IntView, these questions are derived from abstract criteria by using the Goal-Question-Metric method (GQM) [16], which defines metrics for a set of quality dimensions in products, processes and resources. The subject matter expert has to answer a question that refers to the abstract criterion completeness: "Are all aspects of the qualification need covered in order to be able to define the qualification topics? " However, the question for the human factors expert, referring to the same criterion and the same product is quite different: "Are all characteristics of the target group covered which are required to design the user interface of the courseware? " The validity of these scenarios has been tested in interviews with representatives of each role and by applying the criteria in case studies.

The errors and problems that are discovered during the inspection are collected and discussed in an inspection meeting. Errors are corrected and, if necessary,

another inspection cycle might be started. Hands on experience with this inspection method shows that the participation of the different roles at every stage of the project facilitates an agreement about the current development of the product upon the project members. A summary of the activities that are performed during a perspective based inspection is given in Table 2.

Table 2: Description of activities to be performed during the perspective based inspection.

	Activity	Description
1	Distribution of documents	The quality manager distributes the output artifacts of the previous phase and the description of the product that is to be verified to all representatives of the roles that participate in the verification process along with the checklists and tailored scenarios.
2	Reading the descriptions	All participants have to read the descriptions of the product that is verified. They answer the checklist and scenario questions and document errors or problems. Checklists and error lists are sent back to the quality manager.
3	Preparation of verification meeting	The quality manager collects annotations, discovered errors and problems and distributes the complete list to all participants.
4	Verification meeting	The verification list is discussed. The participants decide which errors and problems require further actions and who is in charge to resolve them. If required another verification cycle is scheduled.
5	Post processing of verification meeting	Those who are in charge for resolving errors revise the documents and products and report the changes to the quality manager who checks these improvements (and starts the new verification process, if applicable).

Table 3: Different role categories and respective roles that are involved in the courseware engineering process.

Role category	Role
Management roles	project manager (PM); quality manager (QM)
Pedagogical and Content development roles	instructional courseware designer (ID); subject matter expert (SE); courseware author (CA)
Technical / graphical roles	programmer (P); courseware programmer (CP); human factors expert (HF); multimedia developer (MD); graphic designer and artist (GD)
Customer related roles	customer (C); potential learner (L); expert of the customer (EC)

3.2 Courseware Testing

While the products of the phases 2-10 are verified to assure courseware quality, the products of phases 11-13 are validated by courseware tests. These tests include component tests at different levels of integration (11-12), a system test (13) and a user acceptance test (13). The test cases for these validations are generated during the specification of the requirements (2) respectively during courseware unit design (4) and during the specification of functionality requirements (8).

A test case contains instructions of activities that have to be performed with the courseware as well as expected results or system behavior. Figure 3 shows an example of a test case concerning the minimal configuration requirement. In addition to the test cases, a checklist for the non-functional items of the user interface, the courseware units, and the courseware itself has to be applied. Such non-functional and thus non-testable items are, for instance, graphical elements, their content as well as their position. Therefore, classical tests have to be combined with an inspection of these non-testable elements in order to assure the quality of all four courseware dimensions.

All tests are organized and supervised by the quality manager; they specify the test cases to be applied and compare the test results with the expected results from the test case specification. If there are any deviations, they identify the problems behind the deviations and make arrangements in order to solve them. The quality managers also control the solution of the problems and organize a re-test. A test cycle ends when all test cases run properly without any deviation form the expected test results.

		Instructions	**Expected results**
TS2	NF14	Access the WBT on a computer with minimal configuration: 133 MHz processor; 64 MB RAM; 8 MB graphics; 17" monitor (resolution 1024 x 768); True Color (24 Bit); mouse; bandwidth 56 KB; Netscape Navigator 4.X	First page is displayed without error messages or problems
		Repeat all test scenarios in this test suite	see expected results of the test scenarios

Figure 3: Example of a test case. It refers to a user requirement that concerns the minimal configuration of the computers.

3.3 Prototyping

Prototyping is a quality assurance methodology that is applied in the early phases of a courseware development project. It helps to avoid problems with requirements and to identify problems in the requirements specification or in the design early on. Furthermore, prototyping enables the development of usable courseware at an early stage in the project.

Prototyping can be used to elicit fuzzy requirements or to clarify ambiguous or incomplete requirements in the requirements specification phases (2 and 8) in order to avoid requirements errors [4]. For the first purpose, the main requirements generated by the team should be integrated into the prototype. For the second purpose, the prototype should realize ambiguous or not properly understood requirements that can be clarified only in terms of a prototype [9]. Both kinds of prototypes have to be presented to potential users as well as to the customer. The responses of these people are recorded and analyzed [12]. This can be repeated as long as there are open or unclear requirements that are worth clarifying through a prototype. The results of using prototypes during the requirements specification phase are more stable requirements, which are known better and are closer to the expectations of the potential users and the customer [9].

3.4 Evaluation

In the IntView life-cycle model, evaluations are used to measure the impact on, and the gains for the end users [17], during a validation. Obviously, this requires a running system or at least a running component that can be used by the end users. Nevertheless, this kind of validation is of high importance for courseware since quality criteria like learner performance or usability can only be measured while representatives of the target group are working with the courseware. The products evaluated are the first courseware prototype in the courseware design phase (3), the executable/usable courseware (13) (formative evaluation) and the used courseware (14) (summative evaluation). The design should be evaluated in a prototype implementing the user interface and its functionality as well as a short sample of the courseware content (e.g., a single unit). This prototype should be given to experts as well as to representatives of the target group in order to record their usage of courseware and their opinions on it [8]. The formative evaluation with target users should be conducted preferably in the final delivery and administration environment ([3], [8]). In this evaluation as well as in the summative evaluation, the use of the executable courseware, the opinions of the test users on all views, and the performance of the user in the subject after the use is recorded. The results are used as input for a rework phase, in which the identified errors and problems are resolved and the content of the courseware is improved [15] and evaluated again. An evaluation cycle ends when all participants are satisfied with the product.

4 Preliminary results

The IntView methodology has been applied in several projects to develop different kinds of courseware. These projects revealed that IntView is an efficient tool to control the quality of the produced courseware as well as budgets and time schedules [7]. In particular, the application of the quality assurance methodology showed that:

- performing inspections of the early products revealed many errors and problems, which could be removed with a minimal of effort.
- producing a prototype and evaluating it elicits the expectations and the satisfaction of representatives of the target group very early in the project and allows improvements to the courseware to be made with little further effort.
- the quality of the resulting courseware was already high before the tests. Therefore, the tests could be performed with minimum effort.

To sum up, the application of life-cycle encompassing quality assurance reduced the effort spent on rework of products and the courseware and increased the quality of the resulting courseware. In addition, users answered in a small poll that they are satisfied with the quality of the courseware produced using IntView.

5 Conclusions

The IntView methodology for courseware engineering is filling a gap in current courseware production by enabling projects to control budget and schedule better as well as to produce high quality courseware. In particular, the life-cycle encompassing quality assurance methodology, which is integrated into IntView, allows to assure courseware quality right from project start. It integrates and adapts quality assurance methods from software engineering as well as from courseware development approaches in order to cover all development artifacts and the final courseware as it is proposed in software engineering.

Initial results of applying IntView and in particular its integrated quality assurance methodology in several projects are positive. Nevertheless, we are still working on fine-tuning the methodology and applying it to different domains. We also plan to conduct several empirical studies to get quantitative data on the quality and effectiveness of the methodology.

Acknowledgements

The development of the IntView courseware engineering methodology was partly funded by the "e-Qualification Framework (e-QF)" project under grant 01AK908A of the German Federal Ministry of Education and Research (*BMBF*).

References

[1] U. Becker-Kornstaedt. Der V-Modell Guide: Web-basierte Unterstützung eines Prozeß-Standards. In *Workshop of the GI-Fachgruppe 5.1.1 Vorgehensmodelle – Prozessverbesserung und Qualitätsmanagement, April 20*, 1999. In German.

[2] B. Boehm. *Software Engineering Economics*. Prentice Hall, Upper Saddle River, 1981.

[3] M. Driscoll. *Web-Based Training: Using Technology to Design Adult Learning Experiences*. Jossey-Bass/Pfeiffer, San Francisco, 1998.

[4] A. Endres and H. Rombach. *A Handbook of Software and Systems Engineering: Empirical Observations, Laws and Theories*. Addison-Wesley, New York, 2003.

[5] H. Friedrich, G. Eigler, H. Mandl, W. Schnotz, F. Schott, and N. Seel, editors. *Multimediale Lernumgebungen in der betrieblichen Weiterbildung: Gestaltung, Lernstrategien und Qualitätssicherung*. Luchterhand, Neuwied, 1997. In German.

[6] A. Ginige. Web engineering: Methodologies for developing large and maintainable web based information systems. In *Proceedings of IEEE International Conference on Networking India and the World. Ahmedabad, India, December 9-12, 1998*, pages 89–92, 1998.

[7] I. Grützner, D. Pfahl, and G. Ruhe. Systematic courseware development using an integrated engineering style method. In *Proceedings of the World Congress "Networked Learning in a Global Environment: Challenges and Solutions for Virtual Education, Technical University of Berlin, Germany, May 1 - 4*, 2002.

[8] M. Hannafin and K. Peck. *The design, development, and evaluation of instructional software*. Macmillan Publ., New York, 1988.

[9] P. Jalote. *An integrated approach to software engineering*. Springer, New York, 1997.

[10] O. Laitenberger and J.-M. DeBaud. An encompassing life-cycle centric survey of software inspection. *Journal of Systems and Software*,

50(1), 2000.

[11] W. Lee and D. Owens. *Multimedia-based instructional design: computer-based training, web-based training, distance broadcast training*. Jossey-Bass/Pfeiffer, San Francisco, 2000.

[12] D. Lowe and W. Hall. *Hypermedia & the Web: an engineering approach*. John Wiley & Sons, Chichester, 1999.

[13] C. McCormack and D. Jones. *Building a Web-Based Education System*. Wiley & Sons, New York, 1998.

[14] M. Ochs and D. Pfahl. eLearning Market Potential in the German IT Sector: An explorative study. Technical report, IESE Report No.006.02/E, Kaiserslautern, March 2002.

[15] F. Schanda. *Computer-Lernprogramme: wie damit gelernt wird; wie sie entwickelt werden; was sie im Unternehmen leisten*. Beltz, Weinheim, Basel, 1995. In German.

[16] R. van Solingen and E. Berghout. *The Goal/Question/Metric Method: A Practical Guide for Quality Improvement of Software Development*. McGraw-Hill, London, 1999.

[17] S. Weibelzahl, S. Lippitsch, and G. Weber. Advantages, opportunities, and limits of empirical evaluations: Evaluating adaptive systems. *Künstliche Intelligenz*, 3/02: 17–20, 2002.

[18] C. Weidauer. Ein Vorgehensmodell für die industrielle Entwicklung multimedialer Lehr- und Lernsysteme. In Forschergruppe SofTec NRW, editor, *Studie über softwaretechnische Anforderungen an multimediale Lehr- und Lernsysteme*. September 1999. In German.

Document Information

Title:	Improving Courseware Quality through Life-Cycle Encompassing Quality Assurance
Date:	December 2003
Report:	IESE-081.03/E
Status:	Final
Distribution:	Public