Producing Web Course Material with IBM Knowledge Factory Team

Harri Laine and Teemu Kerola
Department of Computer Science, P.O.Box 26, FIN-00014 University of Helsinki

Harri.Laine@cs.Helsinki.fi

1 Introduction

The Department of CS at University of Helsinki has now for 1.5 years participated in IBM sponsored TUeLIP (Top University e-Learning International Program) project. TUeLIP is a collaborative project of six European Universities (their CS departments) and IBM. Its goal is to produce Web based e-learning material and courses to be used by the participants.

As most participants had scant knowledge of e-learning in general, the project has been a very good kick-off to introduce e-learning aspects in general to the CS departments involved. The expectations on the kind of e-learning to be used varied from self-study courses to co-operative learning, and from interactive Web courses to online instruction sessions. However, the project is not restricted to any specific course type but encourages the participants to try out new techniques and styles in their courses.

The participants were given a free ride in the form of IBM sponsoring the development of one prototype e-learning course. Introduction to Databases was selected as the prototype and its development was distributed among four universities. This gave us and all the responsible universities a good view on IBM's way in producing e-learning courses.

In general, the approach is very professional and thorough, although, it may not be suitable for our courses at large. It is like a well-defined software engineering process as compared to common arts like way of course production. However, understanding this process well gives us valuable insight on how we should proceed with our own material production.

IBM Knowledge Factory (KF) is the content production part of IBM's greater Mindspan Solutions (IBM, 2002a) scheme, that is geared at companies easing themselves into e-learning. The KF team produces, with clients' help, the final product. Knowledge Producer (KP) is an IBM internal software tool that is used by KF team to actually put together the product.

Using the course material is a separate issue to producing it, and in general the e-learning courses could be run on any platform (e.g., WebCT). However, KP is especially geared to produce material well suited for Lotus Learning Space (LLS) (IBM, 2002b). The prototype course was implemented on LLS environment.

We now introduce the Knowledge Factory process with more details. We also briefly describe our experiences with the development process, the material produced and the use of this material. Finally, we summarize our experiences and give some ideas of the future.

2 Knowledge Factory concept

A Knowledge Factory team consists of many people in various roles (IBM, 2000). Some roles can be taken by the client (company or University instructor wanting the final product), but most of the roles are filled with Knowledge Factory people. The roles are divided into two groups: External Team members are specialist and or consultants of various expertise areas, whereas the Core Team members do the actual product development. External Team has 8 roles and the Core Team has 6 roles. In practice, one person can play multiple roles in one production run, and some roles can have many people assigned to them.

The External Team can include roles played by the client. In our case, the Subject Matter Expert was the instructor who had lectured the course earlier and who was also responsible for the new Web based course. Other positions were from IBM. Production Manager has the overall project control. Other expert positions were Media Architect, Technical Specialist,

Audio/Visual Specialist, Production Artist, Photographer, and Quality Tester. Audio/video specialist also found a suitable voice to present all spoken text. In our case, we as the client were also heavily involved in quality testing during the project.

The Core Team is internal to IBM. An Instructional Designer is a specialist in e-learning and (s)he also works as the liaison between the client and the core team. A Courseware Engineer is technically responsible of putting together the final product with various software tools. Content Developers do most of the actual production when they transfer the Detailed Design Documents into final e-learning material with the Knowledge Producer system. Content Analyst keeps the Content Developers in line so that the final product is what the Subject Matter Expert wants. Media Coordinator assists in integrating different medias (graphics, animations, video, audio) and Quality Coordinator checks that all internal and external standards and guidelines are being followed in production phase.

3 Knowledge Factory process

The production process has many phases which each produce some specific document detailing the work done so far. Each phase can be implemented with different people manning the roles described above, and the documents produced in one phase are used as specifications for the subsequent phase.

Before the real production begins, one must analyse current material and goals. This determines what type of material we want to produce. Possible deliverables include Web books, material for collaborative learning, and material for Web lectures. One must also consider whether any electronic performance support system would be used. We must meet student requirements, such as can everyone log in at the same time, or do they all have access to a multimedia lab with high-speed network. All this will finally end up with consensus on what type of e-learning material we want to produce and how that material will be used in actual web course. Many instructional content delivery types can be used together, so that each instructional module is delivered with a method best suitable for it. For example, one can mix and match Web books, Web lectures and stand-alone simulators.

In our case, we decided to produce interactive Web books. One essential requirement in some parts of the course was to have a live database connection.

The first production phase delivers a High Level Design Document (HLDD) for the whole course. It states the overall goal as well as other client expectations for the product. It gives the course outline, look and feel, modules and which order the modules must be studied. It also specifies attributes for the target audience so that the final product will be suitable for it. For example, course size, student nationalities, sex, age, reading level, education level and computer literacy will affect the type of material developed and how it will be used. HLDD locks in menu structures and general navigation style to be used everywhere.

HLDD can specify an instructional theme (E.g., Pizza Restaurant with cartoon style approach) that will then be carried through the whole material. An instructional strategy and performance objectives are specified for each module and each topic. Learning goals determine when students can proceed from one module to the next. A general implementation framework must be agreed upon, so that each designer will know which functionality can be used. For example, whether voice should be used or not. In our case, KP and LLS were going to be used. Thus only the functionality within KP could be asked for.

HLDD is produced in a series of meetings with instructional designer and the client, with lots of help from the experts in the External Team. In our case, the HLDD was 27 pages long.

After HLDD is completed, each module is independently specified as a Detailed Design Document (DDD). The DDD gives a description for each view (page) that the student might encounter during the course. The views are drafted (e.g., with PowerPoint) with enough information for the content developers to produce the final product. Each view contains location information (e.g., module, topic, sub-topic) as well as the instructional information

(e.g., "voice: database is ...", or "hotspot: show animation XYZ here"). Only functionality given in the HLDD framework can be used.

The DDD's are produced in meetings with Instructional Designers and Subject Matter Experts, with other External or Core Team members consulting. The Subject Matter Expert knows what are the instructional goals in each module, and the Instructional Designer does the drafting while all the time considering the limited functionality in the agreed upon framework. Other experts are consulted (e.g., by phone) when needed.

The HLDD and the DDD's are given to content developers and the rest of the core team for actual production. Core teams are a limited resource and one must book them well in advance.

The overall product is then complete, and the External Team Quality Tester will check it before it is given to the client for final evaluation. Problems are then resolved by iteration and/or consulting previous agreements stated in the HLDD and DDD's.

Finally, the product is ready to be used either at client's own environment, or at some IBM server accessible to clients and their students. In our case, the course server was installed in IBM's LLS server.

Although this process may seem large and cumbersome to university instructors accustomed to designing and implementing their own course material alone, we found it to be manageable in size as well as easily adjustable to our own specific needs.

4 Experiences with producing the course material

Introduction to Databases was selected as the pilot course for the TUeLIP project. The course was divided into 18 modules. These were distributed among 4 universities. We in Helsinki were responsible for 5 modules including Relational Algebra and three modules on SQL. We were ready to proceed fast, thus our first module became the pilot for the pilot. Dividing the responsibility of the course among many persons caused problems on the schedule. Soon it became evident that all 18 modules could not be prepared by the deadline. Actually only 8 modules has been implemented by now, and how and when to continue is still open.

We started with the Relational Algebra module. This is traditionally lectured in two hours accompanied with a practice session and home works. The starting point for making the module was the Finnish manuscript material (mainly in MS-Word format) made for the relational algebra part of the Introduction to Databases web course in Helsinki. This course material had been developed during spring and summer 2001 and the first courses using that material were to take place in autumn 2001. Thus we did not have any experience of the use of this material in a Web course. We did not know what kind of input was expected for the KF process. However, we decided to translate our Finnish manuscript in English and made only minor changes on the contents. Some examples and tasks for students were slightly modified. The manuscript contained textual parts, drafts of images, and outlines for animations to be included to illustrate the relational algebra operations. There were many mathematical formulas in the material, and it used a lecture like pedagogic style. The topics were first explained and then practiced.

The transformation of the MS-Word manuscript into a collection of HTML-pages had been straightforward in our Finnish web course. The formulas were transformed into images, and the text was split on web pages so that each relational algebra operation had a page of its own. Pop-ups and links were implemented as defined in the manuscript. Basically it was just a technical transformation. A technical writer in our university did the transformation and negotiated with the author when necessary.

The KF process used in building the TUeLIP module was different. First we spent a lot of time in specifying what should be the outcome of the course/module development effort. The specification concentrated on higher level issues, goals and motivations. Actually, most of this specification should have been done once for the entire course, but now it was done

for each course module. We found out that the development tool placed restrictions on the presentation of the course material. The material was to be split into small non-scrollable screens, basically comparable to PowerPoint slides.

There were two attempts to build this module. The first one followed the manuscript strictly. The design task was, in principle, to split the material into a sequence of pages. To keep the students active, additional reflection questions were added into the material. Instructional designers were requesting for some general theme or background story for the module and proposed the Pizza taxi company that was used in some examples of the manuscript. We had to change some examples of the manuscript to fit with the theme.

The design was outlined as PowerPoint slides. The slides contained all material intended for the page and comments and instructions for subsequent phases of development. We started the design as a session lead by the Instructional Designer. She was not familiar with the topic. During the two day session we designed the HDDL, the overall structure of the course and about 20 DDD course pages. The Instructional Designer copied material from the manuscript, pasted it on PowerPoint slides, negotiated with us and wrote down instructions. After working for a few hours this way we came to a conclusion that we could do the splitting of material much faster by ourselves, especially because this module had many structurally similar parts and we had already covered two of them. So we continued on our own. After we had split the material into slides and included some additional tasks, the Instructional Designer took over the material, which at that time consisted of about 150 slides and 30 pages of animation scripts. The material was intended not only to replace lectures but also the time spent on reading text book, doing homework and attending practice sessions. After still one full day session and many e-mails and phone calls the final design was produced. It consisted of about 150 slides but the proposed animations were reduced to sequences of steps. Graphics were not present and it was impossible to see how to navigate through the course material.

A brief demo that contained the explanation of the example database and one relational algebra operation was ready in October 2001. The demo was fine. The amount of resources used in developing the demo module affected resource allocation for later modules. Furthermore, the number of modules to be developed was reduced.

The first delivery of the pilot module on the Lotus Learning Space environment took place in January 2002. Most of the module was fine, but there were a lot of small bugs, especially in formulas, and navigation did not work as we had assumed. Query building part of the module was a disaster. The theme (pizza taxi) and the example database were overemphasized.

In January 2002 another KF team took over the module. Database experts who knew also about the topic were included in the course development teams. The new team had new ideas. They proposed changing the pedagogic style to induction learning (Holland et al., 1986) so that the students would have to deduce the operations and the formulas based on examples presented to them. Originally, the operations and their formulas were introduced first and practiced after that. The style was changed and we had to figure out new examples for introducing the concepts In the first version only some pages contained audio. Now it was included in all pages. Graphics and navigation were redesigned. New decorative graphics were inserted. A second delivery took place in March 2002. We had asked for browser and operating system independency, but it became obvious that Internet Explorer in Windows environment was the only browser that could be used. There were still some navigational and some other minor problems. No animations were included and the conclusion part of the module had some problems. But mostly it was OK, and this was the module to be tested in experimental courses.

We made also another TUeLIP module, about SQL data definition. This module was simpler than the relational algebra module. It did not use induction learning, but used a more traditional approach instead. The first delivery in April 2002 was satisfactory.

Other universities prepared six modules during spring 2002. They did not have as much material prepared in advance as we had, and there were some problems in fitting their sched-

ules and resources with the KF team. In spite of distributed development the modules fit together quite well.

5 Experiences with the material

It became obvious that only a few modules of all the designed ones would be ready during spring 2002. However, there was urgent need to experiment with the modules during spring or summer 2002. It was decided that the universities involved would arrange courses, where parts of the course would be based on completed TUeLIP modules and the rest would be covered by lectures or by some other means. In Helsinki we experimented with a course for foreign students in May. The course had 6 hours of lectures, and four TUeLIP modules were used, two of which were our own. In addition, we translated the SQL-part of our Finnish web course into English. The translation was done in order to prepare for the future TUeLIP modules assigned to us. Because the look and feel of the SQL-modules was not the one used in the TUeLIP modules, we called them pre-TUeLIP modules. This part of the course used live database through the SQL-Trainer (Laine, 2002) practicing tool. We consider to include this tool also to the TUeLIP versions of these modules, although it cannot be linked fluently with LLS.

There were 15 students registered for the course, but only 11 students ever logged in the Lotus Learning Space server, and only 9 students were logged in the SQL-Trainer server in the latter part of the course. Finally, 8 students took the exam and all of them passed the exam. At the same time there was an independent exam for the same course and we used the same questions for both exams. Independent exam was taken by 17 students and 10 of them passed it. However, we cannot draw any far-reaching conclusions about these statistics. The exam had only one question about the topics taught in the four TUeLIP modules. Also, independent exams are typically taken by students that have failed to pass the course earlier. There are also students that try to pass the course without practicing at all. Some of the students that passed the independent exam had studied the course using the Finnish web course material.

Students reported some technical problems in the use of the Lotus Learning Space. Strict untold assumptions about the Windows setting caused problems for some students. Discussion groups did not work for all students. Nobody succeeded in providing feedback through LLS forms even if they tried. However, students were mostly satisfied with the material. Some considered audio as unnecessary, but others liked it. The tasks in TUeLIP modules were considered easy as compared to the SQL-Trainer tasks that they had to pass to get credit.

6 Summary and future plans

Knowledge Factory process is a well-defined process. It has many roles and phases. Tasks and roles in the IBM's process reflect good understanding of how to build an e-learning course. We feel this model suits well for companies doing rather small e-learning courses without having people dedicated for education. However, university courses are typically large. The subject matter experts are also experts on teaching the matter. We do not exactly know how much developing of these pilot modules cost. But our guess is that developing university courses this way is far out of our economic possibilities. To find out proper ways to combine the roles, tasks and phases, and to find proper people for those roles is essential to develop a more cost effective way of course building. Fully utilizing the capabilities of teachers and providing them support when needed might be the solution. To provide support for instructors our department has accepted an e-learning strategy that defines two support units: a pedagogic support unit and a technical support unit.

To complete the Introduction to Databases course one possibility is that universities try to use the KP tool by themselves to build the still missing modules of the prototype. We will also experiment with other content production systems in near future.

References

Holland, J. H., Holyoak, R. E., Thagard, P. R., 1986. Induction: Process of inference, learning and discovery. MIT Press, Cambridge, MA.

IBM, 2000. Ibm knowledge factory. CD-ROM.

IBM, 2002a. IBM Mindspan Solutions.
URL `http://www.ibm.com/mindspan`

IBM, 2002b. Lotus learning space.
URL `http://www.lotus.com/learningspace`

Laine, H., 2002. SQL-Trainer. In: Proc. of the 1st Annual Finnish/Baltic Sea Conference on Computer Science Education. University of Joensuu, Department of Computer Science, pp. 13–17, report A-2002-1.