Integrating Real Practical Experience in ICT Education

E.O. de Brock
Faculty of Management and Organization
University of Groningen
P.O. Box 800
9700 AV Groningen, The Netherlands
e.o.de.brock@bdk.rug.nl

ABSTRACT

An important challenge for advancing the quality of education in Information and Communication Technology (ICT) is to integrate education and practical experience. It is sensible to start such active learning already in an early stage of the curriculum, in a valuable, real-life, and effective manner. It is desirable to reach that goal in such a durable way that we can easily account for the continuous changes and extensions in the underlying ICT itself. This paper presents our experiences in integrating ICT education and practice from an educational designer's point of view. We will give several hints on how to design such an integrated curriculum module. We first explicate which educational "design questions" deserve attention in designing such a module. For each such design question we present the solution we chose, some specific examples, our experiences over the years (gradually improving the organization of such a module), and alternative solutions there might be. We also give some quantitative information.

Keywords: Practical experience, integration in education, course projects, educational design, knowledge versus skills, project management, active learning

1. INTRODUCTION

A well known and important challenge of education in Information and Communication Technology (ICT) is to integrate education and practice; see for instance (Harris, 1994). The importance of practical experience in IS education, for instance, is beyond any doubt; see among others (Rollier, 1993) and its references. It is sensible to start such active learning already in an early stage of the curriculum (as also follows indirectly from (Daigle, 1998) and (Serva, 1999), for instance). It is a challenge to accomplish this in a valuable, real-life, and effective manner (Harris, 1994; Saiedian, 1992). It is desirable to reach that goal in such a durable way that we can easily account for the continuous changes, advances, and extensions in the underlying ICT itself (Reed Doke, 1999). In this paper, which is an extended version of (de Brock, 2000), we describe our experiences to integrate ICT education and practice from an educational designer's point of view.

We first present the main design questions we address (and answer) in this paper. We can distinguish between *educational questions* (internal questions) and *questions relating to practice* (external questions). We start with the main design questions relating to practice.

- What are suitable projects?
- How do we get suitable projects from practice?
- Which appointments do we make with our customers?
- Which parties are involved?
- What must each party do?
- How do these parties communicate?

We continue with the main educational design questions.

- How do we check the progress of the projects?
- How do we judge the results?
- In which sensible way can we take project management aspects into account in the projects?
- How do we take the project management aspects into account in our judgements?
- How can we account for the continuous changes in the underlying ICT-platforms?

2. CONTEXT

The main running example in this paper is our curriculum module *System Development (in) Practice*, in the third trimester of the second year in the variant *Information Technology* of our 4-year curriculum *Technical Management Science* at the Faculty of Management and Organization at the University of Groningen. (In this business school a year is divided in three trimesters and a trimester consists of 14 weeks.) This module contains basic technical skills such as systems analysis and design skills and database design and implementation skills, as well as non-technical skills, such as teamwork, project management, and communications skills, which are very important in practice (Van Slyke, 1998), usually regardless of job classification (Reed Doke, 1999). Although our illustrations sometimes contain some module-specific details, we will in general concentrate on the more generic issues that are relevant to any "educational designer" in the IS-area.

In our SD-module the students have to develop (and install) a small information system, including the delivery of technical and user documentation, for some real external customer who actually needs such a system. The students start from "scratch", i.e., a (usually vague) problem description of about one page, written by that customer. So the students will experience a practical project "from the very beginning to the bitter end". Each student has 160 hours available for this module. The students have to work in groups. This year (in Spring 2000) we started with 34 students divided into 9 groups, working on 8 customer prolems. (To one of the customer problems we assigned two "competing" development teams.) Last year we started with 25 students divided into 7 groups (working on 7 customer problems).

In the preceding trimester the students have already followed the courses *Databases* and *System Development Theory*, which gave the students the prerequisite background. E.g., the course *Databases* treats topics such as (formal) data modeling, constraints, queries, views, transactions, and translation of these items to SQL, following (de Brock, 1995). It also pays attention to system documentation. Groups of two students also have to formalize, implement, and document an informally but clearly described case in a DBMS (currently Oracle). Usually, a group consists of one of our (MIS) students and a student from computer science. The course is offered jointly to MIS-students and computer science students; see (Bock, 1999) for the advantages.

The course *System Development Theory* treats topics such as information analysis, user participation, interviews, prototyping, various forms of system development, method engineering, testing, and project management. The students also have to perform an interview (where it is the instructor who acts like a user).
The current division in modules has evolved from our experiences with earlier versions in the course of years. For example, before the current division in *SD Theory* and *SD Practice* we had the courses *System Development 1* and *System Development 2*, each with some theory as well as some practice. However, since the practical components were relatively small and short-term and did not determine the course result completely, we could not really enforce a strict project-like situation.

In (the third trimester of) the third year our curriculum contains a module called *PA/IT* (*Practical Applications of IT*). This module has a similar practical focus but sometimes chooses alternative solutions for the design questions we address. Here each student has 120 hours available. The students have matured one more year and in this module the starting point can be any problem - except potential ones for *SD Practice* - that relates ICT and the business of the customer in an integrated manner. By "integrated manner" we mean here that the business problem and the ICT problem should not be two separate subproblems but need to be "interwoven", i.e., you cannot solve them independently from each other (even though the business problem will be leading). A typical example is the problem of selecting a software package for a given company: determining company specific selection criteria and weight factors, scanning the market, judging the products, and determining the preferred product (or a short list).

The topics of interest can (and actually did) change over the years. In 1995 and 1996, for example, we combined the PA/IT assignments for some groups with an assignment they had for their management accounting (MA) course. For MA each of these groups had to describe the "as-is" (or IST) situation regarding some MA issue within one given company, and for PA/IT they had to write an advisory report that worked out a possible, ICT-enhanced "as-could-be" (or SOLL) situation regarding that same MA issue. In 1998 some of the groups had to judge a specific service level agreement for midrange systems that a given company wanted to sign with a software house. This year (in Spring 2000) the module was thematic: Each group had to make a strategic advisory report as well as an illustrative pilot e-site for some medium-sized company that wants to start up its own internet

activities; the course started with some specific lectures and intensive practical exercises.

3. DESIGN QUESTIONS AND POSSIBLE ANSWERS

We detail our main "design questions" from the Introduction and present the solutions we chose, some concrete examples, our experiences, and sometimes some alternative solutions.

3.1 EXTERNAL QUESTIONS

We start with the questions relating to practice (external questions).

- *What are suitable projects?*
  E.g., suitable kinds, topics, well-definedness, and size?

Given the purpose of this module, any request to develop a small information system for some external customer on any topic is suitable, in principle. Any other incoming problem relating ICT and the business of the customer (in an integrated manner) can be passed on to the responsible colleague of our module *PA/IT* (see the last paragraph of the Introduction). We do not presuppose any special background knowledge of the particular application area from our students.

We ask our "prospect" to deliver some written problem description. Such a description ought to be (and usually is) at most one page. For example, this year the shortest one was half a page and the largest one was three pages (including contextual, organizational and some additional information).

We subsequently judge whether the problem has (or can be adapted to) *roughly* the proper size. The requirements to actually install the developed software at the customer's site and to include the technical and user documentation as well are usually missing, but we add them also. Agreeing upon the *precise* boundaries of the information system with the customer is part of the students' management of the project (like in practice)! This can usually be negotiated by leaving out (or adding) some applications or queries.

In order to give an indication of sizes, we mention some statistics of the projects of this year: the resulting databases contained on average 10 tables, with a minimum of 4 tables (with 52 attributes the highest attribute density) and a maximum of 22 tables (with 70 attributes the lowest attribute density), and contained on average 55 attributes, with a minimum of 30 and a maximum of 99 attributes.

- *How do we get suitable projects from practice? E.g., via the students themselves, the colleagues, we as instructors ourselves? When do we start searching? (Not too soon, not too late?)*

In the beginning we really had to "spread the word" and search for projects. By now many people (former students, colleagues, contacts outside) are aware of this possibility and contact us or point this out to other people, who then contact us. Community service and volunteer organizations are suitable domains; see also (Rathswohl, 2000). Moreover, we already see the new students in the trimester before and tell them that they can bring in proposals themselves (e.g., from their students' club or sports club). This also aids their motivation.

Proposals can come in any moment of the year but have to wait for execution until the third trimester. (Usually, an alleged urgency of the proposal is not that hard.) Proposals for our module *PA/IT* can be executed whenever students are available.

- *Which appointments do we make with our customers?*
  E.g., what about financial appointments? And what quality/gain/delivery time can we guarantee them and what not?

As far as financial appointments are concerned: we do it for free! This leaves us the necessary freedom to waive quality guarantees (e.g., in case of badly functioning project teams), to negotiate on the boundaries of the system, and, in extreme cases, to cancel a project team. Of course other appointments could be made here. We tell the customers that the quality can be worse *or* better than that of "professionals" (which is actually true). As an alternative, the same problem can be tackled by different project teams in parallel. (For instance, this year we assigned two "competing" development teams to a customer who was somewhat worried about a guaranteed" good end result.) What we do guarantee them though is that they will get at least a very good insight into their problem.

The delivery time is usually 3 to 4 months (roughly the length of the trimester), but can be negotiated.

- *Which parties are involved?*
  E.g., customers, students, instructors, student assistants, others? What is the role of the instructor(s) and of the student assistant(s)?
  Is there an "account manager" role? By whom?

We can confirm the project management rule that there should not be too many parties involved, and the role of each party should be clear. There are at least the customer(s), the developers (i.e., the students), and the account manager (i.e., the instructor) with their classical roles. We note that the communicator/ proposer of the project need not be one of the customers. This must then become clear to all parties. After a while the communicator does not play an important role anymore in such cases.

Student assistants can play a useful role in helping the students with technical questions concerning software and hardware and in assisting the account manager, monitoring the progress of the project teams, checking the deliverables (timeliness and rough contents), and assisting with other time-consuming tasks. However, the responsibility remains with the instructor.

- *What must each party do?*
  E.g., which tasks and roles must each party have? Who does which tasks? And what makes a good project team size? Do we need one instructor or maybe more?

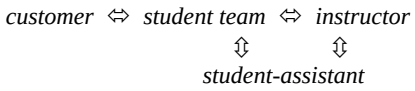
We just answered most of the questions on roles and tasks above. The distribution of the roles and tasks within a team has to be made explicit by the students themselves (preferably beforehand but, since that is difficult for the students, at least afterwards). In our experience a good project team size turns out to be three or four students. This is small enough to prevent individual students to "retreat in silence" and large enough to experience the problems of working in a team. (A few years ago we allowed the students to choose a team size themselves, given the customer problems. As an experiment we proposed five stdents per team, but nevertheless they still preferred four.)

In principle, the students can choose from the proposed projects any project they like. However, the instructor (in his role of account manager) has to take care of a fair distribution of team members over projects. Here, the instructor can also take the personal (technical as well as non-technical) skills of his students into account. Depending on the number of students and the number of proposed projects, it is possible that *no* team or *more than one* team is assigned to a project. The latter situation can lead to an interesting competition among those teams!

It is our experience that a team of one instructor with one student assistant can be a very effective and efficient combination. But it is a lot of work. However, working with more than one instructor has the danger of measuring by different standards. Special attention should then be given to this point. Sharing the same student assistants helps but is not enough.

- *How do these parties communicate?*
  E.g., who communicates with whom? And how?
  (Face-to-face conversation, telephone, ocuments, fax, snail mail, e-mail, Web?)

During the project, the main communication lines are as depicted in the figure below. Before the start of the project and near the end of the project, i.e. during the final demonstration of the (almost) final system, the instructor discusses the project (result) with the customer. During the project the instructor can occasionally ask the customers about their experiences with the teams.



The communication is primarily by face-to-face conversation, phone, documents, and e-mail. E.g., as instructor I received this trimester roughly hundred emails from my students and roughly fifty from my student assistants. I sent roughly thirty emails to the student teams and only a few to my student assistants; most of the emails I sent were replies. The students are obliged to read their email regularly (which they usually already did anyway).

Proper and timely communication with the customer is an explicit responsibility of the student teams. They should make appointments about that with the customer. This aspect is a typical part of the real practical experience! Also the student teams should ask the customer about his absence/presence during the project period (vacation, etc.) and account for that in their plans. (For example, one of the customers had his holidays of four weeks during the three month project period in Spring.)

It is our experience that the internal communication within some groups is not very good. Nevertheless, we consider the internal communication as *their* responsibility. When we accidentally communicate something to only one of the team members then that member has to inform the others (if necessary). It would also be a good idea that students mention by default the other team members under "CC" in all the emails they send.

3.2 INTERNAL QUESTIONS

We continue with the educational questions (internal questions).

- *How do we check the progress of the projects?*
  E.g., which milestones, intermediate, concept and final products do we want to see (minimally)?

The first thing the students have to deliver is (a first version of) a plan, for which we set a (latest) delivery date, typically within one week. Often the first version of their plan contains some clear flaws; in that case they have to deliver a second version (within a few days).

The plan has to contain at least a deadline for the whole project and explicit deadlines for a concept version (not necessarily bound or in full color yet) as well as for a final version of the technical and user documentation. The students have to distinguish these two "external" versions from all kinds of intermediate internal versions. The plan should also contain a latest date for the demonstration of the system by the students to the instructor, student assistants, and customer together (preferably at the customer's site). This demonstration should take place shortly before the delivery of the final version of the product. The system should already contain some sample data before the demo begins. Being the foundation of the information system to be developed, the underlying conceptual or logical database model (i.e., structure/schema and constraints) is also a very useful intermediate product, for checking progress as well as content. Other intermediate products, milestones, or deadlines depend more or less on the phasing chosen by the student team. However, we should see or hear from each project team at least once in two weeks. Therefore, the plan might need to contain intermediate progress meetings as well. The instructor can use the meetings with a team to discuss the progress of the project in relation to the plan, the contacts and appointments with the customer, the internal communication, possible bottlenecks in the project, next phase(s), and project-specific attention points. Project-specific attention points can be: global architecture of the system, the technically more complex applications, determination of the precise boundaries of the system, and interfaces with other information systems. Examples of pretty complex applications (for them) in this trimester were: matching clients (for optimal "fitting" of the clients of a "human networking" agency), reading out the registries of all computers in a network (for a system administrator's application), and providing a tunable, generic information system for questionnaires (and their answers) that were continuously under development (for a field study of an academic researcher).

In general, the instructor can use all meetings with the students to relate their situation and experiences very concretely to the lessons taught (but not yet always learned) in previous courses, in order to aid a deeper understanding of the real meaning of the theory (Rollier, 1993).

All group appointments such as progress meetings and demonstration have to be arranged by the students themselves (which is often less straightforward than they expect).

In our case, the final product consists of the technical and user documentation and the installed software at the customer's site, as well as a copy of the software and documentation for us.

- *How do we judge the results?*
  E.g., per project team or per individual student?
  Refined marks/grades or just "sufficient/not sufficient"?

Since system development is a "team sport" in practice, we judge the results per project team and not per individual student (except for very exceptional cases). The wide variety in the given project proposals as well as in the solutions of the teams make it quite hard to compare the teams and to determine refined marks/grades. Moreover, a few years ago (with a slightly different setting though) the team results were of such different levels that we should have used a logarithmic scale to fit in the classical A-F or 10-1 range. We only give the judgment "sufficient" or "not sufficient" (which is sufficient). Last year one group result was "not sufficient", this year all results were "sufficient".

As written earlier, the team size is small enough to prevent individual students to "retreat in silence". The teams also have to make the task division among their members explicit. Moreover we regularly talk with the students. Like an oral exam, this also shows us whether their individual involvement is sufficient.

- *In which sensible way can we take project management (PM) aspects into account in the projects?*
  E.g., the PM aspects planning, task division among project members, and limitation of the project. For instance, do the customers or instructors make the decisions on these aspects, or the students? Are the students already able to make good decisions on these aspects? And what if not? What is the role of budgets?
  Money or time budgets? Fixed budgets?
  How do we obtain the proper learning effects here? And what about additional learning effects which come from team-specific experiences?

Decisions on PM aspects such as planning, task

division among project members, communication, and (exact) limitation of the project are made by the students, although we check them regularly on their reasonability. As an example, students generally tend to underestimate the time needed for a good implementation. With MS Access, for instance, they often are already acquainted with some simple standard solutions to some simple standard problems. However, non-standard solutions or non-standard problems turn out to be very hard, cumbersome, and/or laborious in Access. They often experience this for the first time in these projects (which require more than only simple standard solutions to some simple standard problems). Usually our students do have a good idea of the time needed for documentation, practically thanks to a case in our course Databases and theoretically by our course System Development Theory.

Since the students are usually not yet able to make good planning decisions, they can propose to adapt the plan during the project, within bounds and especially informing us *in time* (!), which we defined as "before the deadline concerned passes". We return to this later on.

Since we do the projects for free and each student has 160 hours available for this module, we work with time budgets instead of money budgets. Therefore we can treat the projects as fixed (time) budget projects. Given the fact that the instructor is responsible for the rough size of the project as well as for the team size, it is subsequently the responsibility of the students to control their time budget in more detail. (Recall that the students can negotiate the *precise* boundaries of the system with the customer.) The learning effects are very direct in this way: the "victims" of bad budget control are the students themselves.

Team-specific experiences can lead to interesting additional learning effects. (Strike the educational iron while it is hot.) Last year, for instance, one of teams (that also just started its own IT company) developed a solution in their project that was correct but very hard to maintain. Nevertheless, they did not see any real problems here, at first. The problems became suddenly clear for them, however, when we said: "Imagine that this solution was meant as standard software your company wants to sell and maintain, with several versions in the future".

A striking team-specific experience this year was the moment when it turned out that one of the teams had to sign a secrecy agreement with a basic fine clause of 25,000 guilders (more than 10,000 dollars), excluding additional civil liability claims. After some deliberation with us, one of the students called his lawyer – yes, he had one – who explained that such agreements are not unusual in business. He advised the student to agree upon some kind of clearance statement of the customer at the end of the project. When the customer agreed, the students signed and went on. Indeed, a learning effect they will never forget.

- *How do we take the project management aspects into account in our judgments?*
  E.g., do they influence the final mark or grade, and to what extent ?Can the project team (or an individual student) be dismissed by the instructor? How can we justify this?

Most PM aspects directly pay back to the students by their influence on the amount of work and the quality of the work of the students, and hence indirectly on the final judgment. Justified by the fact that communication and planning are usually "deadly" important in IT projects, a project team (or an individual student, in extreme cases) can eventually be dismissed by the instructor if the team does not inform us in time on proposed adoptions of the plan. As said before, we define *in time* as "before the deadline concerned passes". If a deadline passes without adoption, the team gets a warning (a "yellow card", like in some sports). If the team gets two more warnings (an "orange card" and finally a "red card"), the team is excluded from further participation and none of the team members gets a grade for the module that year. Of course, at the start of the module these "rules of the game" have to be communicated very clearly to the students. In our two-year experience with these rules, several teams received a yellow card, only very few got an orange card, and no team ran into a red card. Last year (in Spring 1999) one student received an (individual) red card and this year one student stopped in an early stage after receiving an individual yellow card.

- *How can we account for the continuous changes in the underlying ICT-platforms?*
  E.g., does the instructor prescribe a fixed platform, one the students are already acquainted with? *Or do the students and customers decide on this in mutual consultation? What if the students are not yet acquainted with (parts of) the underlying technology? What about the availability of the platform?*

The students and customers decide on the underlying ICT-platform (i.e., OS, DBMS, and perhaps a network) in mutual consultation. If the students within a certain team are not yet acquainted with (parts of) the underlying technology, a different composition of the teams might help (in a very early stage). Otherwise, (at least) one of the team members has to get ac-

quainted with that technology (within reasonable bounds). In their (time) budgets, the students are allowed to consider this as part of the project itself. (We also point out that this should usually not hold for the money budget in practice.)
On the other hand, we also made a short dedicated manual of a currently popular DBMS (which is now Microsoft Access, but a few years ago Dbase IV and later Dbase V).

As a consequence of the mutual consultation, availability of the platform (hardware as well as software) is usually not much of a problem in our experience. Customer, student or, hardly necessary, the university are the available options.

4. CONCLUDING REMARKS

Of course it is very valuable to evaluate the module with the students and student assistants. Such evaluations can lead to successive improvements for the following year. We mention some possible improvements in our case.

- Testing their products, as part of the project, seems to get too little attention by the students. It would be an elegant educational measure (at least from the instructor's point of view) to let each team test the product of another team as well; e.g., team 1 tests project 2, team 2 tests project 3, … etc., and the last team tests the first project.
- We only have a few general meetings with all the students at the beginning of the trimester (in order to startup). However, it would be useful for the students to have a general meeting with all other students during the project period as well, in order to exchange experiences, problems, and solutions.
- It is educationally useful (though time-consuming) to attend at least one information analysis session with the customer per student team, and to evaluate it afterwards.

Each year, evaluation of the module shows that the students judged the projects very valuable and a really good learning experience. They implicitely confirm (Rollier, 1993) that such a practical case is necessary for a deeper understanding of the real meaning of the theory.

In our effort to advance the quality of ICT education and training, we succeeded to integrate our education with practice in an early stage of the curriculum in a valuable, real-life, and effective manner. We reached that goal in such a durable way that we could easily account for the continuous changes and extensions in the underlying ICT itself.

5. ACKNOWLEDGMENTS

The author is grateful to his (former) students and student assistants for their constructive comments over the years. The author is also grateful to the reviewers for their helpful suggestions for improvements of the presentation of the paper.

6. REFERENCES

Bock, D.B., et al., 1999, "Integrating Computer Science and Information Systems." SIGCSE Bulletin, Vol. 31, No. 4, pp. 56-60

Daigle, R.J. and M.V. Doran, 1998, "Facilitating Bloom's Level One through Active Learning and Collaboration." Journal of Information Systems Education, Vol. 9, No. 3

de Brock, E.O., 1995, Foundations of Semantic Databases. Prentice Hall International Series in Computer Science, Hemel Hempstead; 230 pages

de Brock, E.O., 2000, "Integrating IT Education and the World Outside." In Proceedings IRMA-conference, Idea Group Publishing, Hershey (PA), pp. 863-864

Harris, A.L., 1994, "Developing the Systems Project Course." Journal of Information Systems Education, Vol. 6, No. 4

Rathswohl, E.J., 2000, "Using Community Service-learning in an Information Systems Course." In Proceedings IRMA-conference, Idea Group Publishing, Hershey (PA), pp. 868-869

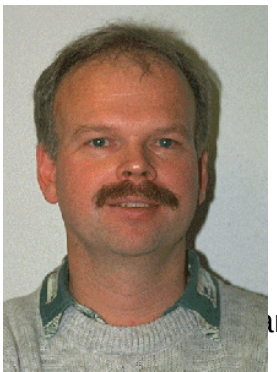
Reed Doke, E. and S. Rebstock Williams, 1999, "Knowledge and Skill Requirements for Information Systems Professionals: An Exploratory Study." Journal of Information Systems Education, Vol. 10, No. 1, pp. 10-18

Rollier, B., 1993, "The Database Project: Maximizing its Value." Journal of Information Systems Education, Vol. 5, No. 2

Saiedian, H., 1992, "Guidelines for a Practical Approach to the Database Management Systems Course." Journal of Information Systems Education, Vol. 4, No. 1, pp. 23-29

Serva, M.A. and M.A. Fuller, 1999, "Teaching Evaluation: Acknowledging the New Realities in the Modern Business School Classroom." Journal of Information Systems Education, Vol. 10, No. 1

Van Slyke, C., M. Kittner, and P. Cheney, 1998, "Skill Requirements for Entry-Level IS Graduates: A Report from Industry." Journal of Information Systems Education, Vol. 9, No. 3



Dr. Bert de Brock is an

associate professor of Information Technology at the University of Groningen since 1993. He received a M.Sc. in Mathematics at the University of Groningen in 1979 and a Ph.D. in Computing Science at the University of Technology in Eindhoven in 1984. From 1985 to 1990 he worked at Philips Research on the PRISMA-project (Parallel Inference and Storage Machine) and the ECHO-project (Electronic Case Handling in Offices). In 1990 he and one of his former colleagues started a company in the areas of IT-consultancy, post-academic education, and analysis, design, and construction of (tailor-made) information systems for customers. His research interests include databases and information systems. He is the author of the book *Foundations of Semantic Databases* (Prentice Hall International Series in Computer Science, 1995).

Information Systems & Computing
Academic Professionals

STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

ISSN 1055-3096