Teaching Portfolio

Dr. Per Runeson

Software Engineering Research Group
Dept. of Communication Systems
Lund University

v 1.3

1. Introduction

Born 1966, growing up with “Hej matematik” at school, with mother and grandmother being teachers, two older sisters being teachers, and since 14 years married to a teacher, it is hard to avoid teaching. However, I tried my best in the choice of education program, when I began at the Lund Institute of Technology, LTH, in 1987 at the program for Computer Science and Engineering.

After taking my degree 1991, I was employed at Q-Labs, as consulting expert in software engineering. However, I gradually moved towards Ph.D. studies and graduated 1998 with a thesis on how to measure faults and failures in software inspection and testing. On graduation, I was offered the opportunity to assume responsibility for developing the software engineering bachelor's program at the LTH Engineering School in Helsingborg. Under relatively free circumstances, I designed a three-year program in the domain of software engineering, and was the program director responsible for coordinating the program during its first three years.

Hence, my background is a mixture of industry and academia, giving insight in different kinds of learning, both for myself, and for others which I have been a teacher.

2. Personal Viewpoint on Teaching

My personal view on teaching is that it should focus on student learning [1]. As a teacher, I am not satisfied by “bringing out” my message, but measure success in terms of how much is “getting in”. The same holds for communication in general, but is even more important to stress in the teacher-student relation, which is unbalanced in favour of the teacher.

In a pedagogic course during summer 2001[1], we were asked to find a picture representing one's view of teaching. My choice was a glass sculpture by Bertil Vallien, see Figure 1. It represents to me the teacher shaping the student. The student is in focus, and the teacher has the possibility to form the student to some extent.

Figure 1. “Head” by Bertil Vallien.

Later in the course, we met the basic theories by Fox [3], see Table 1. I still like the shaping role, but guided by his model, I tend to turn towards the growing student with the teacher as a gardener. This theory also emphasizes the individuality of the student. Each student will learn their own way, not necessary the well structured and thought through structure of the lectures and courses which they attend.

The shaping model is particularly valid for the supervision of post-graduate students. With Ph.D. supervision, you may follow the development of the student during a period of 4-5 years, rather than the 7 or 14 weeks for normal courses at LTH. During the long period, the interaction between the student and the supervisor develops. As the student develops, gradually more responsibilities and more challenging tasks can be given to match the student at the specific point of time.

1. Sommarinstitutet 2001, http://www.hgur.se

From my point of view, the students are the "customers" of the university. This means that the students should be in the focus of the learning situation, but it does *not* imply that the "customers" are always right. The teacher has a better long-term perspective than the student, but the student is often better at seeing what is successful within a shorter range of time.

TABLE 1. Basic theories of teaching, after Fox [3], with slightly modified terminology

	Simple theories	**Developed theories**
Subject focus	Transfer	Travel
Student focus	Shape	Grow

My topic, software engineering, is defined as "the application of a systematic, disciplined, quantifyable approach to the development, operation, and maintenance of software; that is, the application of engineering to software" [4]. Hence it is natural for me to take a holistic viewpoint of the topic and to focus more on the complete picture than specific details of the topic. I prefer structuring the topic top-down, connecting to the student's existing experience and knowledge, and then support the way towards deepened knowledge and skills. However, the learning process is not strictly linear, but rather iterative. Hence, I strive towards first giving a complete picture, on a very high level of abstraction, adding some details on lower levels, end then returning back to the complete picture again, now with a little more details in it, and then iterate again. From the student point of view, this means that they have a context in terms of a "roadmap" to relate more specific knowledge to. This procedure is illustrated in the teaching examples (Section 3.1 and Section 3.2) and is related to the Kolb circle, see Figure 2.

In the process of relating to knowledge, I think that practising is a very important contributor. As the human has five different senses, it is waste of bandwidth, just to use the ears. There is Chinese proverb which I rephrase as:

What you hear, you will forget
What you see, you will remember
What you do, you will know

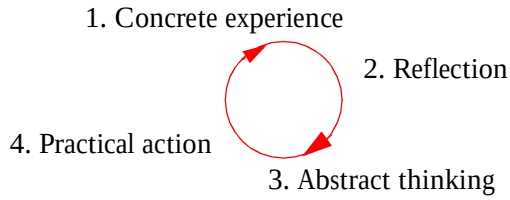

Figure 2. The Kolb circle [5]

As a consequence, teaching should include project work. In addition to practising a topic, projects involves practicing communication skills and develop other relational skills. The course projects have, in my opinion, *not* to be complete software projects, with real customers and requirements derived from them. Instead, I prefer projects which are tailored to specific learning goals. This focuses learning on the specific issues, which are the topics of the course. In the analysis of the projects, reflection and generalization in terms of abstract thinking should be stimulated.

Expressed in terms of Bloom's taxonomy (Table 2), course projects support the students in reaching the *application level*, and the reflection and generalization performed during and after the projects, support reaching the *analysis level*. A sequence of projects may support reaching the *synthesis level*, while I think that the *evaluation level* cannot be reached before having practical experience from software engineering in industry, or in a post-graduate program.

TABLE 2. Bloom's taxonomy [2]

6	Evaluation
5	Synthesis
4	Analysis
3	Application
2	Comprehension
1	Knowledge

This discussion leads to the goals of the education, goals for education programs, goals for courses and goals for specific course items. When setting up the software engineering bachelor's program (see Section 3.1 and [6]), the division of goals in *knowledge, skills and attitude goals* have been very helpful. Knowledge refers to, for example, basic terminology and concepts of the topic, which should not be lost in the strive for deeper understanding. Skills include the application of the gained knowledge; moving the knowledge "from head to hand". Attitudes refer to the motivation behind how one is acting in different situations. It is questioned whether it is a task of the teaching to impact on the students' attitudes. My point of view is that the attitudes are always impacted on implicitly, and then it is better to make them explicit, to enable students to question attitudes. Returning back to the initial "shaping" model, the students are always changed, and the most fair approach is to report openly our intentions with the shaping.

Finally, some words on examination. Ideally, learning should be driven by curiosity and a never-ending search for more knowledge and experience. However, the world is not ideal, but needs quality monitoring and control. The examination should act as monitoring of the learning process. When we need this monitoring, we should strive towards making the examination as such, a learning occasion as well. In my courses I prefer examination through project assignments, reports and presentations. Being very much in favour of "alternative" examination methods, I have to remind myself that written exams are still an option, in particular for examination related to knowledge goals. Project assignments are better suited to examine skills objectives and report writing are well suited to follow up attitude goals.

To summarize, teaching...

- is student focused, shaping or gardening the growth of the student;
- relates to a complete roadmap, in which practising is important for learning;
- is goal oriented, and examination means assessment related to goals.

3. Teaching Examples

3.1 SE bachelors Program

The most extensive teaching experience I have, at least in terms of the scope, lies outside the lecture rooms. I have planned and supported the implementation of the software engineering bachelor's programme at the LTH Engineering School in Helsingborg[1]. The key values behind the program are:

- The students should achieve a holistic view of software development. Important attitudes are customer-focused development, and that software development is a long chain from idea to final product. Hence the students should be given a broad overview from the beginning, which also provides an identity of the education program chosen.
- In order to provide a comprehensible overview of the development over the three years, each year is given a certain focus; *individual, project and organization* focus. This is referred to as the program policy and helps students see the intentions of the three years.
- To ensure a balance between different topics, five main areas are defined to help bridge the large gap between the programme and its goals as a whole, and courses and their goals on the lower level. The main areas are *Mathematics, Computer Science, Software Engineering, Hardware and Systems Engineering, and Non-technical issues*.

1. http://www.programvaruteknik.hbg.lth.se

In the program, the Kolb cycle (Section 2) is iterated in many cycles, each cycle adding something new to the knowledge base. The software engineering courses of the first year (Section 3.2) are more focused on the knowledge and comprehension levels of Bloom's taxonomy. The project focus of the second year is dedicated to application and analysis, and we reach the synthesis level in a course on Software Quality in the third year.

The development of the program was very much goal driven. Goals were set up in terms of knowledge, skills and attitudes [6]:

- "The knowledge goal is that the students acquire basic and deepened knowledge within the areas of software development, computer science, mathematics and mathematical statistics, computer and systems engineering and non-technical areas such as oral and written communication, economics and English.
- The skills goal is that the students be able to join a large industrial software development project and after a short period of time be productive in the project. The students also be able to develop and vitalize an employer's way of conducting and managing software development projects.
- The attitudes goal is that the students achieve good understanding of industrial software development, and thereby understanding for the need of a balance between technology and methodology."

Based on these goals, a brainstorming meeting was held with colleagues to define a set of courses to be given to the students to achieve the goals. The courses were grouped into the course areas, and lined up in sequences of courses that built on each other. The procedure was iterated, as the first set of courses fitted a master's program rather than a bachelor's.

The program was launched in the fall semester of 1998. In my role as an education program director, I was responsible for the overall structure, and in the role as a teacher, I gave courses, among those the introductory course (Section 3.2). The combination of teacher and program director was important, as the courses provided meeting points with the students which gave feedback from the students on how the program worked, or did not work.

The structures in terms of goals and course groups have been valuable in the communication with the students, to motivate them to take a course by setting it into the context of the whole program. Furthermore, it has been helpful in the communication with the education program board, as the discussions have been held on a higher level of abstraction than the course definition level. Changes to the program took place during the first year, based on direct student feedback, see Section 3.5, and courses were exchanged later which did not achieve the educational goals, see [7].

During summer 2001, the program leadership was handed over to a newly employed colleague. The hand-over seems to have been successful, and one of the contributing factors was that the new program director also took the responsibility for the introductory course.

3.2 Introductory Course for SE

In the software engineering program, there is an introductory course which aims to give an overview of the whole area of software engineering. The purpose was to provide a "roadmap" that the students could use for their "navigation" through the education program, hence directed towards knowledge and comprehension [2].

The course is designed with a set of traditional lectures and training sessions, primarily connected in pairs, where the training session applies the topics presented in the lecture. Each pair of lecture and training session is approximately related to one course later in the education program. However, to provide an integrated view of the area and the program, "the Software Tapestry" was introduced. This is a visual view of the different steps in a software development project, see Figure 3.

Figure 3. The Software Tapestry

Another step towards integration in the course is the dictionary. The students are given a list of terms each week, which they are to explain. The terms are, by the way, given in both Swedish and English to enable the students have relevant terminology in both languages. After each week, the continuously growing dictionary is reviewed, either by a student peer, or by a teacher.

Finally, a project is run following the steps in “the Software Tapestry”, to give some practical experience. The steps are, of course, not followed in detail, but in a very overview fashion, to provide a first insight to what is to come during the education program, and during the professional life.

The course has been given in different versions since 1998. In the second run, a parallel course on communication and reading skills was coordinated with this course. A sales brochure was to be developed and presented by the students, for the product that was developed in the project. The coordination did not require much effort from the teacher’s side, but nevertheless provided a more holistic view from the student’s perspective. Unfortunately, this is not possible any more due to a changed program schedule.

Further developments of the course concepts would be to return to “the Software Tapestry” in later courses in the program, and thereby integrate the courses better. Having changed responsibilities, this is currently out of my scope, but there is still a development potential in the concept, see Section 4.2.

3.3 Introductory Course for InfoCom

In the education program for Information and Communication Technology, I have developed an introductory course in software engineering, with a substantial part of project management issues. As this course is given during the first year, I have taken the opportunity to include some course issues aimed at “soft” skills, and attitudes towards software engineering. The course is linked to a subsequent course given by another department. A simple requirements specification for a chat system is derived in this course which is partially implemented in the subsequent course. This interface between the courses and departments is not very extensive, but helps the students integrate the knowledge and skills gained in the two courses into a complete picture.

One specific part I of the course is the training sessions around the question *What is an engineer?* It was inspired by the picture task, which we performed at the pedagogic course, referred to in Section 2. At the very first lecture, the students are asked to find a picture which tells them something about what an engineer is. The students choose a wide variety of pictures, including electronic equipment, people, screws and bolts, “kylskåpsingenjör Stig-Helmer Olsson” and computers, see Figure 4.

As the very last task of the course, they turn back to the picture, and write a short report about their picture and how their view was changed during the course. Again, the variety among the students was large. One student wrote a single line that “his view of an engineer was his father, being an engineer, and the view was not changed during the course”. At the other end of the spectrum, the view of the engineer had changed from being rather individually oriented with a technical focus, to a more team-oriented view. See further [8].

Figure 4. Sample pictures chosen.

The main result of the tasks was to catalyze reflections and discussions about the role of an engineer. The picture helps thinking in new directions, and involves more capabilities than just reading and writing skills.

3.4 Project-based Courses

In the courses I have been teaching, project assignments have mostly been a part. Within the software engineering domain, some teachers state that only complete projects with real customers are sufficient to give the experience needed on project work.

I do not agree with this statement. By limiting projects in scope, the learning can be focused on specific issues, and making the project environment sufficiently complex to give students hands-on experience of all the issues, which are in focus for the current course. By climbing the ladder of Bloom's taxonomy for different sub-topics, the synthesis can be achieved in a later step, like the example presented in Section 3.1.

One example project based course is the flagship for our department, “PUSS” (ProgramvaruUtveckling för Stora System), which has been run for at least 10 years in different shapes, and has been continuously evolved by different teachers [10]. In the course, the same products is developed over and over again, by new groups of students; project groups of 15-18 students. Still, the challenges of large software development projects are illustrated during the course. The students meet the challenges of communicating to and informing so many people, being dependent on each other's work etc. The technical problems are rather limited, and those are not the focus for the course. The problems lie in effectuating the project.

The “PUSS” course focuses on application - learning by doing. During the project work, a structured approach to system development is fostered, i.e. turing software development into engineering. At the end of the project, a final report is delivered as an important part of the assessment, which comprises analysis of what happened during the course. Before, one report was written for the whole project, which implied that the project leader s learned the most. As an improvement step, we have introduced an individual analysis report for each project member, to foster analysis and to constitute the basis for individual examination.

Another project I have used in teaching, with limited scope, is the one in a course regarding Software Verification, which I was responsible for developing. This course covers one step in the middle of the development cycle. The students are given a product, which is developed through the steps up to the verification step. Thereby, they can spend the time in the course, specifically towards the course topics.

In the Software Verification course, the guidelines given to the students are less restrictive compared to the guidelines given “PUSS” course. It turned out that the students felt insecure in the Software Verification course, which also caused more conflicts within the project group. For the next run of the course some more guidelines were given to support the learning in the direction related to the goal of the course, which was primarily software verification, and secondary project management, not vice versa.

In general, the value of projects are that the students experience and feel more directly, and are set into situations where they have to apply the theoretical knowledge, leading to higher level of learning. Again, traversing the Chinese proverb: *What you do, you will know.*

3.5 Student Feed-back

Student feed-back is very important in order to provide better learning situations, both within an instance of a course, and for coming generations of students. The most common tool for feed-back are questionnaires, free or more structured ones, either on paper or web-based. I consider this being a very “blunt” tool. It provides some basic level of assessment, but it has to be complemented with both quicker and deeper methods. In order to achieve the really valuable comments, occasional discussions provide the best feed-back.

I have an example of written feed-back from a course, which illustrates some of the problems with written, quantitative course evaluations. Firstly, the response rate was 42%, and the question is what did the 58% non-answering students think? Secondly, there are no significant deviations from satisfaction. Thirdly, the deviations that seem to be in the data, indicate that the book is among the least appreciated issue, while the project is among the highest graded issue. However, based on qualitative information, we have decided that the book is not changeable, but needs to be used more by the students to give the intended gains, while the project will be given better support. If we were to follow the feed-back results, the project needs no changes, while the book should be replaced.

With the drawbacks of written student feed-back in mind, I prefer direct communication with the students as a means for getting the really valuable feed-back. It is quick, direct, and bi-directional. However, this requires an open and clear relation between the students and the teacher. In my experience, some enablers or catalysts are needed to initiate this communication.

As a program director for the software engineering program, I set up meetings with all students, or with student representatives, but these were attended by a small group only. Instead, when more spontaneous discussions started, these were more fruitful. For example, in the introductory course for InfoCom students (see Section 3.3), we had a session on interpersonal communication models. In the break after this session, a very fruitful discussion regarding the course and the education program took place. The session on communication models acted as a catalyst for the discussion and spending some time during the break was really worthwhile.

4. Directions for the Future

My viewpoints on teaching are summarized as: the teaching...

- is student focused, shaping or gardening the growth of the student;
- relates to a complete roadmap, in which practising is important for learning;
- is goal oriented, and examination means assessment related to goals.

As I am pragmatic and opportunity driven, I prefer to show a direction, and the detailed implementation will be worked out in the specific roles and occasions where I have a chance to impact on the teaching. The areas I specifically want to focus on mean work on different levels, related to the areas above. *Motivate* students, which involves methods to motivate students to learn, rather than forcing them, and fostering learning for life, not for the test. *Integrate* courses into an education program, which involves a strengthened role for the education program director, teacher awareness of other courses and a focus on teaching teams. *Assess* students related to learning goals as a part of learning, and a continuous quality assurance of the learning, rather than an end-product pass or fail check.

4.1 Motivate

Spending effort to motivate students to learn is an investment that pays back in two respects: firstly, it is more satisfactory to work proactively and positively; secondly, I think that joyful learning is better learning. Hence, I want to focus on student motivation for the future.

I think that there are (at least) three important contributors to student motivation:

- Knowing why I as a student should learn this.
- Knowing how it relates to my current knowledge.
- Finding the learning situation joyful.

The first issue, I want to address by even more focusing on the goals; the goals of a course, and the goals of an education program. If the goals are well formulated, they provide a rationale for the course. I always present the goal in an initial lecture, as well as in the course program, but I want to integrate them continuously in the course, trying to break the overall goals down into the constituents of the course, e.g. seminars, projects and tasks, and to return back to the goals several times during the course. I would also like the standard for goal setting on the program and course level to improve.

Figure 5. Goal focus.

Another method is to show where the topic of the course fits into a complete picture, like the one presented in Section 3.2. Knowing where some pieces fit creates motivation by itself.

As a teacher, you expect that the students have grasped what you have presented so far in the course, and in previous courses you have taught. They have not! On the other hand, the students have knowledge and experiences gathered outside the classroom, which can be brought in to help add to the current course. An example of the use of this strategy is the picture task presented in Section 3.3, where the students were asked to bring in what they had before the course, and then at the end of the course reflect on what was changed [8]. For future runs of the course, more focus could be set on the analysis of the outcome.

Teaching an applied engineering topic, I also have the chance to bring in guest lecturers to tell students what the outside world looks like and corroborate their need for the knowledge and skills I am teaching. However, the format for these lectures is not very stimulating, as they often have a one-way communication format. I would like to make the sessions more interactive, and also involve more active work by the students.

Finally, making the learning situation as joyful as possible involves all kinds of variation, but also a good relation between the student and the teacher, based on mutual respect and both parties fulfilling their part of the contract. Here I want to continue finding the balance between being an authority, which I am in my role as a teacher, and being a coach that supports the students in their learning efforts.

4.2 Integrate

During the planning and implementation of the software engineering bachelor's program, I realized that there is a potential for more integrated education programs, see Section 3.1, to give the students a more complete picture and thereby provide more efficient learning.

Let us compare an education program to a software system, e.g. a word processor to an education program, in my case the software engineering program. In the software system there are components, e.g. editor, spell checker, font manager, and in the education system, the components correspond to courses. In a software system, the components shall have well defined *interfaces* to ease communication, be *cohesive* and have as *low coupling* between each

other, to ease the maintenance and understanding of the system, but still focus on the overall goal of the system. This is also very much like a jigsaw, see Figure 6.

Figure 6. Jigsaw.

In the education system, we have to maintain the same rules as in a software system case. The courses should have well defined interfaces in terms of pre-and post-conditions regarding knowledge and skills. The courses should be cohesive, i.e. focus on specific learning goals in contrast to supplying a little of everything. The coupling between the courses, i.e. interdependencies should be avoided from a course maintenance perspective, but this is often in conflict with the overall system goal. Teachers should be aware of the contents of other courses to be able to provide their piece of the jigsaw into the complete picture. Still, for practical reasons, interconnections between courses should be kept low, to enable continuous improvements of each course. If the courses are too much interconnected, a change in one course will lead to changes in many other courses.

I want each study program to have a well defined overall picture, and each course to have well defined interfaces, in terms of pre- and post conditions and goals. This is an important basis for integration. My vision is that more focus is set on the integration between teachers, supporting communication between them, and integration between courses which is manageable (as the examples given in Section 3.2 and Section 3.3). The program director is the role which may contribute and catalyze this integration, and this potential is not fully utilized. As this is a matter of change management within LTH, the integration has to start in-the-small. Successively when trust and good experiences are built up, there is a foundation for tighter integration.

4.3 Assess

The assessment culture at LTH is very much the culture of the written exams and re-exams. I dislike the written exam, from my personal experience as a student, and from the personal experience as a teacher assessing exams. Hence, I wanted to change the assessment culture, at least in my courses. In new courses, I have used projects and report writing as assessment methods rather than written exams.

Figure 7. Assessment as a part of learning.

However, I think that I have gone too far, and lost the motivational part of individual written exams. As mentioned in Section 2, the written exams have their role, in particular to assess knowledge and comprehension level goals. Therefore, I want to find this balance between individual and group assessment, between written and oral exams, between knowledge, skills and attitudes goals. As an example, we are currently introducing a written exam in the introductory course presented in Section 3.3. To communicate goals via the assessment, we have introduced a model-based assessment approach [9].

For the future, I am also interested in trying some kind of portfolio assessment, as these methods activate the student, moving the “proof burden” to the student to show that he/she has achieved the goals. This is a large cultural step, which has to be handled with care, but I think it is possible to introduce this kind of techniques gradually, to empower the students and improve learning.

5. Summary

Teaching portfolio is a somewhat misleading title for what rather should be entitled *learning* portfolio. My basic standpoint is that what takes place in and between the students is more important, than how the teacher acts. However, the teacher may facilitate learning and provide positive learning environments.

The teaching examples are given to illustrate how the basic principles of my attitudes towards teaching and learning, are put into practice. They are not always as student focus as I wish, but I think they are steps in a right direction.

For the future, my key words are complete pictures and a variety of methods. Students learn differently, and we have to support them in different ways. But they all have to know why they are learning something, and hence we should for this sake strive towards bringing knowledge in its full context.

6. References

1. R. B. Barr and J. Tagg, “From Teaching to Learning - A New Paradigm for Undergraduate Education”, *Change*, November/December 1995.
2. B. S. Bloom, M. D. Engelhart, E. J. Furst, G. F. Hill and D. Krathwohl, *Taxonomy of educational objectives: The cognitive domain*, New York: McKay 1956.
3. D. Fox, “Personal Theories of Teaching”, *Studies in Higher Education*, Vol. 8, No. 2, 1983.
4. IEEE Std 610.12-1990, *IEEE Standard Glossary of Software Engineering Terminology.*
5. D. A. Kolb, *Experiential Learning. Experience as The Source of Learning and Development*, New Jersey, Prentice-Hall, 1984.
6. P. Runeson, “A New Software Engineering Programme - Structure and Initial Experiences”, *Proceeding 13th Conference on Software Engineering Education & Training*, Austin, Texas, USA, pp. 223-232, 2000.
7. P. Runeson, “Experience from Teaching PSP for Freshmen” *Proceedings 14th Conference on Software Engineering Education & Training*, Charlotte, North Carolina, USA, pp. 98-107, 2001.
8. P. Runeson and T. Thelin, “Addressing Attitudes Explicitly in Engineering Education-An Exercise to Stimulate Reflection through Pictures”, *LTH pedagogisk inspirationskonferens*, pp. 40-41, May 27, 2003. Available at http://serg.telecom.lth.se/research/publications/docs/268_SE_attitudes_final.pdf
9. P. Runeson and B. Regnell, “Model-Based Course Assessment - Principles and Practice”, *LTH pedagogisk inspirationskonferens*, May 27, 2004. Available at http://serg.telecom.lth.se/research/publications/docs/269_Model-Based%20Assessment.pdf
10. C. Wohlin, “Meeting the Challenge of Large Scale Software Development in an Educational Environment”, *Proceedings 10th Conference on Software Engineering Education & Training*, pp. 40-52, Virginia Beach, Virginia, USA, 1997.