Learning Probabilities in Computer Engineering by Using a Competency – and Problem–Based Approach

Ahmed Khoumsi and Brahim Hadjou

University of Sherbrooke

1. Introduction

Several studies have detected serious gaps between the objectives of university engineering programs and the needs of an evolutionary economy [1]. As a solution, our department has undertaken a fundamental and major reform of its Bachelor of Electrical and Computer Engineering degrees [2]. The new adopted learning approach is based on *competence development* for *solving problems* and *realizing design projects*.

In this article, after an introduction to the new learning method, we illustrate its application by a problem-based learning (PBL) unit which aims at learning *probabilities* and the use of concepts (such as *Laplace transforms* and *Generating functions (or z-transform)*) that can help for the computation of probabilistic parameters. This PBL unit is denoted APP_{prob}[1]. This paper is aimed essentially at persons interested in new approaches for learning and using probabilities in computer engineering. One of its main contributions is that it demonstrates that PBL can be just as effective as a way of learning theoretical subjects. Note that the PBL approach described here has also been applied for learning of many other subjects. For example, [3] presents a PBL unit for learning compiler design.

This paper is structured as follows. In Section 2, we introduce the pedagogical approach adopted in our department. Sections 3 and 4 present APP_{prob} as an illustration of the PBL approach. In Section 3, we present the competencies, the necessary knowledge for developing such competencies, and the book used as resource in APP_{prob}. Section 4 presents the contextual problem to be solved and the various pedagogical activities realized in APP_{prob}. In Section 5, we explain how students are assessed in APP_{prob}. Section 6 discusses the advantages of the PBL in learning probabilities in computer engineering. And in Section 7, we conclude the paper.

2. Pedagogical approach and organization

In this section, we present the main principles of the major reform of our electrical and computer engineering programs. A more detailed presentation can be found in [2]. Such reform aims at making the objectives of university engineering programs compatible with the needs of economy and society [1].

2.1 Competency and knowledge

"Conventional" engineering programs give priority to knowledge acquisition. With the reform, priority is given to the development of *competencies*. Competency can be seen as an ability to *act* and use *resources*, for solving a given task. Competency is not synonymous with know-how, because the competency is flexible and adaptable, and cannot be reduced to an algorithm. Competency concerns more heuristics than algorithms. In our reformed programs, competencies are classified in four types: scientific and technical competencies, design competencies, interpersonal competencies, and intra-personal competencies.

Development (or implementation) of a competency requires acquisition of knowledge, which can be considered as resources. Knowledge has been classified into three types: declarative (know factual information), procedural (know *how to use* factual information), and conditional (know *when and where to use* factual information). In the context of our engineering program, factual information can consist of, for example, a definition, a theorem, a hypothesis, a rule, or an algorithm.

2.2 Organization of a trimester, PBL approach

The programs are organized around four-month periods which, for simplicity, will be called *trimester*[2]. The programs last eight *academic trimesters*, alternating with four *training trimesters* beginning after the third academic trimester[3]. Each trimester is based on a theme (e.g., computer systems, analog control) and includes two types of activities: six consecutive two-week problem-based learning (PBL) units, and a design project spread over the whole trimester (see Fig. 1). The project is worth 3 credits in each of the first six trimesters, and 6 credits in each of last two trimesters, with a total of 15 credits per trimester.

ABSTRACT

Our department has redesigned its electrical and computer engineering programs by adopting a learning methodology based on competence development, problem solving, and the realization of design projects. In this article, we show how this pedagogical approach has been successfully used for learning probabilities and their application to computer systems.

KEYWORDS

Problem-based Learning, Competence development, Knowledge acquisition, Assessment, Probabilities, Queuing systems, Performance analysis.

1 APP is the French acronym for *Apprentissage par Problèmes et par Projets*.

2 In French, we would say *quadrimestre*.

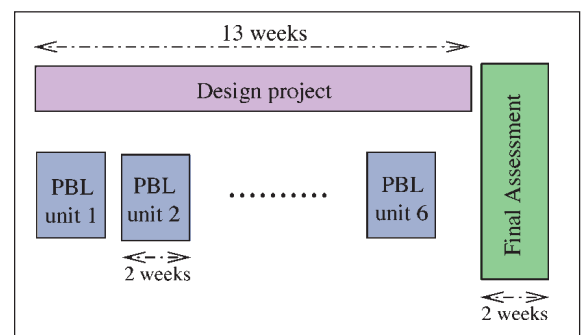3 Henceforth, the term *trimester* means *academic trimester*.

Figure 1: Trimester Structure

Each of the six two-week PBL units of a trimester is based on a problem to be solved, rather than on a discipline or subject as in a conventional program. This approach is motivated by the fact that PBL is the natural mode of knowledge acquisition and competence development. A problem must, not only come from a real engineering situation, but also be presented in such a way that the students have to identify their existing (i.e., previously acquired) knowledge and the new (i.e., not still acquired) knowledge, that are necessary for solving the problem. The formulation of the problem must also lead the students to identify the necessary skills for solving the problem effectively. This learning *contextualization* provides realistic situations where knowledge is applied, and thus, encourages a better understanding of that knowledge.

PBL encourages *active* learning, and thus, students are more *responsible* and *autonomous* in the learning process. Professors are "resources" that react by providing opinions or indications, validating or invalidating solutions, asking questions, etc. But professors should never provide a solution (or information allowing to deduce straightforwardly a solution).

Let us consider Trimester 3 of the computer engineering program, the theme of which is *Computer System Architectures*. One of its six PBL units, denoted APP_{prob}, aims at learning probabilities and their applications in computer systems. As an illustration of the PBL approach, APP_{prob} is presented in detail in Sections 3 and 4. It is worth noting that most of the students in APP_{prob} learn probabilities for the *first* time.

3. Competence and knowledge in APP_{prob}

In this section, we present the competencies aimed in APP_{prob}, the necessary knowledge for developing such competencies, and the book used as resource.

3.1 Competencies aimed in APP_{prob}

The following four competencies have been identified:

3.1.1 C_1: to compute probabilities of events and of random variables

Concerning *events*, the aimed ability is to compute probabilities of occurrences of events. Concerning *random variables*, let us distinguish *discrete* random variables (DRV) and *continuous* random variables (CRV). For DRV, the aimed ability is to compute probabilities that a DRV takes given values or falls within given countable sets. For CRV, the aimed ability is to compute probabilities that a

CRV falls within given sets, e.g., intervals or union of intervals.

3.1.2 C_2: to compute parameters of a probabilistic model

Given a DRV or CRV that models a given stochastic process, the aim is to be able to compute standard parameters such as: *mean value* (also called *expected value*), *variance*, *standard deviation*, *moments* (1st, 2nd, 3rd … orders), and *distribution function*. Concerning CRV, another aimed ability is to compute *density function*.

3.1.3 C_3: to select and determine a probabilistic model for a system

Given a stochastic process described intuitively, the aimed ability is to model such a process by random variable(s) that possibly (but not necessarily) correspond(s) to standard models. For example, in the discrete case we may have Bernoulli, Binomial, Poisson, or Uniform DRV, and in the continuous case we may have Exponential or Uniform CRV.

3.1.4 C_4: to analyze a system by using probabilistic models, and interpret results

The first ability to be developed is to analyze quantitatively a concrete system by using probabilistic models of stochastic processes, and the other ability is to interpret the obtained results. As an example of system, let us consider a simple queuing system that receives and processes requests that arrive at a given rate. The considered queuing system consists of a FIFO queue and a server (see Fig. 2). The FIFO queue stores requests at their arrival, while the server takes requests from the queue and processes them.

Queuing systems have been classified according to several characteristics such as: request arrival law, service time distribution, whether queue length is finite or not, number of servers, etc. We will consider two standard classes: M/M/1 and M/G/1. The aim is then to analyze the performance of a queuing system, through a probabilistic study of the following five variables that can be considered as *random variables*: service time, i.e., the time for processing a request by the server; waiting time, i.e., the time a request spends in the queue before service begins; stay time, i.e., the total time a request
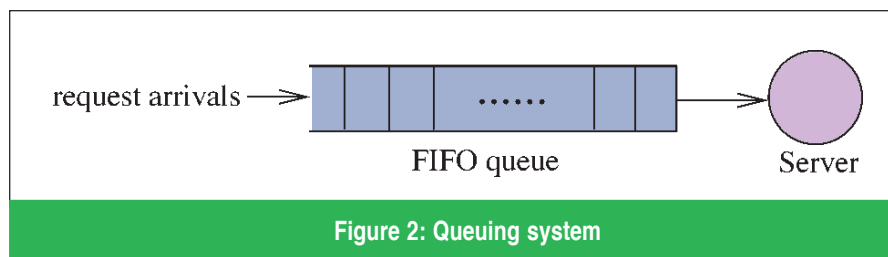

Figure 2: Queuing system

spends in the queuing system (queue & server); number of requests waiting in the queue; and number of requests in the queuing system (queue & server). In this context, competency C_4 can be described by several abilities, such as:

1. to determine which class of queuing system can model a given process;

2. to determine if the queuing system is steady, i.e., the mean number of requests waiting in the queue is finite;

3. to compute the mean values and variances of the above-mentioned five variables;

4. to compute probabilities related to the five variables.

Note that items 3 and 4 are also considered in Competencies 1 and 2 (see Sections 3.1.1 and 3.1.2). The nuance is that in Competencies 1 and 2, general methods are used for computation, whereas in Competency 4, formulae applicable to specific queuing systems can also be used.

3.2 Necessary knowledge in APP_{prob}

Development of competencies of Section 3.1 requires acquisition of knowledge, which has been classified into three types: declarative, procedural, and conditional (see Section 2.1).

3.2.1 Declarative knowledge

Declarative knowledge consists of factual information to be known. We have identified the following list of declarative knowledge elements:

- Discrete and continuous probabilities: event probabilities, conditional probabilities, random variables, expected (mean) value, variance, moment of any order, density function, and distribution function.

- Standard discrete random variables: Bernoulli, Binomial, Uniform, Poisson.

- Standard continuous random variables: Exponential, Uniform.

- Laplace transform.

- Generating function (similar to z-transform).

- Dirac impulse function.

- Simple queuing systems: Markovian (M/M/1) and semi-Markovian (M/G/1)

- Variables considered for evaluating performance of a queuing system: service time, waiting time, stay time, number of requests in the queue, number of requests in the system.

- Performance parameters of a queuing system: such as, mean values of the above five

variables, and probabilities related to them.

Other declarative knowledge is necessary (e.g., derivatives, integrals, series), but is assumed already acquired.

3.2.2 Procedural knowledge

Procedural knowledge is to know *how* to use factual information. We have deduced the following elements of procedural knowledge from the above list of declarative knowledge elements.

- To compute probabilities of events and random variables.

- To compute probability parameters using general formulae applicable for any DRV or CRV. Parameters considered are: mean value, variance, moment of any order, density function, distribution function.

- To compute probability parameters using specific formulae applicable for some standard discrete random variables (DRV), such as Bernoulli, Binomial, Uniform, Poisson.

- To compute probability parameters using specific formulae applicable for some standard continuous random variables (CRV), such as Exponential, Uniform.

- To compute probability parameters of CRV using Laplace transform.

- To computing probability parameters of DRV using Generating function.

- To use Dirac impulse function in order to consider a DRV like a CRV.

- To model a computer system by a Markovian (M/M/1) or semi-Markovian (M/G/1) queuing system.

- To compute mean values and probabilities of the following variables in a /M/M/1 or M/G/1 queuing system: service time, waiting time, stay time, number of requests in the queue, number of requests in the system.

Other procedural knowledge is necessary (e.g., computation of derivatives, integrals and series), but is assumed already acquired.

3.2.3 Conditional knowledge

Conditional knowledge is to know *when* and *where* to use factual information. In our case, the aim is to determine which model to use for solving a given problem, among several models. More precisely, we have identified the following conditional knowledge elements:

- To select the appropriate probability model to describe a computer system.

- To select the appropriate queueing model to describe a computer system.

3.3 Book

A book that we found appropriate for APP_{prob} is [4]. It is complete and adequately structured for APP_{prob}. Notations of APP_{prob} have thus been taken from this book.

4. Pedagogical activities in APP_{prob}

In Section 3, we presented: the objective of APP_{prob}, which is to develop certain competencies, the necessary knowledge that must be acquired for achieving this objective, and the book used as resource. In the present section, we present pedagogical activities that have been elaborated to reach this objective.

Organization of activities of APP_{prob} results from a slight adaptation of a generic organization elaborated by our department [2] and inspired from [5]. The adaptation is essentially due to the mathematical aspect of APP_{prob}, which requires more theoretical work and less practical work, in comparison to other PBL units. A typical organization of APP_{prob} is illustrated in Table 1, where black zones are related to project activities. Let us detail APP_{prob} activities in the following subsections. Note that in addition to several activities under supervision, students occupy the rest of their time with personal study.

Monday-1	Tuesday-1	Wednesday-1	Thursday-1	Friday-1
Tutorial−1 (1h30)	Personal study	problem solving procedures (6h)	Collaboration for problem solving (3h)	Project
Personal study			Personal study	Personal study
		Personal study		

Week 1

Monday-2	Tuesday-2	Wednesday-2	Thursday-2	Friday-2
problem solving procedures (3h)	problem solving validation (3h)	Tutorial−2 (1h30)	Project	Personal study
Personal study	Personal study	Personal study	Personal study (formative assessment)	Sommative assessment (2h)
			Consultation (1h)	

Week 2

Table 1. Activities of APP_{prob}.

4.1 Monday−1: Tutorial−1, problem to solve

For each group (comprising about 10 students), APP_{prob} starts by a 90-minute tutorial meeting (denoted Tutorial-1). Through a collaborative work and under tutor guidance, students:[2]

1. read the terms of the problem to solve, keep only the relevant terms, and formulate succinctly the problem; (*30 minutes*)

2. propose solution alternatives (i.e., tasks for solving the problem) and, for each solution alternative, identify pertinent knowledge (acquired previously or to be acquired); (*45 minutes*)

3. organize and prioritize solution alternatives; (*10 minutes*)

4. review the list of knowledge to be acquired. (*5 minutes*)

The tutor's role in Tutorial-1 consists essentially of asking relevant questions, validating students' prior knowledge, ensuring that learning needs and solution alternatives are well identified. But the tutor *never* presents solutions to the problem. For the sake of clarity, let us introduce the problem that has been used this year (2004) in APP_{prob}:

4.1.1 Problem to solve

You are an engineer in a company which has just signed a contract with a small airline. The latter is growing rapidly and its clientèle is increasing constantly. This small airline uses a single server to register all sight reservations of its flights. It asks your company to study the performance of its server for deciding if and when it must be replaced by a more efficient server. More precisely, the airline needs a performance parameter evaluation of the system to be analyzed as a function of its physical parameters.

Your boss is full of hope to land other contracts similar to the one signed with the airline. He commissions you to realize this study by developing a general and rigorous method for performance analysis of computer systems using a single server. He advises you strongly to heed Mazaya's advice, an experienced colleague in performance analysis but who has not time to realize by herself the task which is asked to you.

After a succinct research, you learn that criteria generally used for evaluating a server performance, are relative to necessary time for processing requests and to necessary memory for storing

requests waiting for processing. You also learn that stability must be guaranteed and that probabilities and queuing models are generally used for this type of study.

A meeting is organized with Mazaya, where you let her know about what you have learnt in your brief research. Mazaya agrees with your learning and urges you to determine the probability models and the mean values of the following five variables: service time of a request (*s*), separating the beginning and the end of the request processing; waiting time of a request (*q*), separating the request arrival at the server and the beginning of its processing by the server; stay time of a request (*w*), separating the request arrival at the server and the end of its processing by the server; the number of requests waiting for service (*Nq*); and the number of requests staying in the server (*N*), waiting or being processed.

Continuing your study according to the recommendations of Mazaya and your boss, you observe that in order to make a rigorous performance study, it is necessary to have the following models: a model for request arrivals, that is, a law that governs the succession of request arrivals at the server; and a model for the server, that is, a law that governs service time. You discuss about that with Mazaya who advises you to model the succession of request arrivals by a Poisson process with rate λ (rate of request arrivals) and to determine under which conditions this model is applicable.

After a more elaborated research, you note that in certain cases, Laplace transforms (LT) and Generating functions (GF) can be very practical for computing probabilities. You consult Mazaya who proposes that you proceed as follows. First, you compute the density function (F_{dens}) and the distribution function (F_{dist}) of *s*. Then, you can compute the mean values of *s*, *q*, *w*, *Nq*, and *N*. Then, you compute the LTs of F_{dens}s of *q* and *w*, and the GF of *N*. And finally, you should compute Prob[N=i], for i=0, 1, 2. She advises you to use the software tool MatLab for computing Prob[N=i].

To determine a model for the server, you ask Mazaya to explain you how the server works. She informs you that requests are processed in the order of their arrivals and that, for processing each request, the server executes the following two operations: a specific treatment the duration of which can be estimated by a constant δ and then an access to a disk in order to add, modify or remove data. She tells you that the disk rotates at a constant speed *r* (in turn/sec) and has a number **k** of sectors, and that a single head is used for access to disk. Every disk access allows to read or write a block consisting of a constant number b of sectors. The last recommendation Mazaya gives you, is to interpret results of your study. More precisely, you have to study succinctly influence of

parameters *k*, *b*, *r*, δ and λ, on the stability, on the mean values of *s*, *q*, *w*, *Nq*, *N* and on Prob[N=i] for i=0, 1, 2.

4.1.2 Results of Tutorial-1

At the end of Tutorial-1:

- The problem is formulated succinctly in something like:
  To use probabilities and queuing models to develop a rigorous method for analyzing performance of a computer system comprising a single server.

- The knowledge identified as necessary for solving the problem, must be close to the list of Section 3.2.

- Students must agree on an organized list of tasks (so-called *solutions alternatives*). This list must look like:

1. To model the request arrivals by a Poisson process, and determine the conditions under which this model is applicable.

2. To compute the density function and the distribution function of service time (*s*).

3. To compute the mean values of: service time (*s*), waiting time (*q*), and stay time (*w*).

4. To compute the mean values of: the number of requests in the queue (*Nq*), and the number of requests in the system (*N*).

5. To compute the Laplace transforms of the density functions of: waiting time, stay time.

6. To compute the Generating function of *N*.

7. To compute Prob[*N*=0], Prob[*N*=1], and Prob[*N*=2] using MatLab.

8. To study the influence of parameters *k*, *b*, *r*, δ and λ on the stability of the system.

9. To study the influence of parameters *k*, *b*, *r*, δ and λ:
   - on the mean values of *s*, *b*, *w*, *Nq*, *N*, and
   - on Prob[*N*=i] for i=0, 1, 2.

Note that the use of MatLab is a prior knowledge that has been acquired in a previous trimester.

4.2 Wednesday-1: Problem-solving procedures

Under tutor supervision, students apply knowledge acquired in personal study, by practicing problem-solving procedures in two 3-hour sessions. This activity consists of solving several exercises and aims at practicing of:

- computing probabilities of mutually exclusive events;

- computing probabilities of independent events;
- computing conditional probabilities;
- computing parameters of discrete and continuous random variables: mean value, variance, moments of 1st and 2nd orders;
- selecting appropriate standard discrete and continuous random variables for modeling given concrete systems;
- computing performance parameters of Markovian and semi-Markovian queuing systems;
- selecting appropriate (Markovian or semi-Markovian) queuing systems for modeling given concrete systems;
- using Dirac impulse function for representing the density function of a DRV.

Exercises for this activity have been carefully elaborated for practicing relevant knowledge to solve Items 1 to 4 and Item 8 of the solution alternatives (Section 4.1.2). In this activity, exercises are solved by students and presented by them to their peers. The tutor validates the presented solutions, but (s)he must not present solutions.

4.3 Thursday-1: Collaboration for solving the problem

Through a 3-hour session and under tutor guidance, students use knowledge acquired so far (in supervised activities and in personal study), and collaborate to elaborate solutions to the problem. After having practiced problem-solving procedures of Wednesday-1, students should be able to solve at least Items 1-4 and Item 8 of the solution alternatives identified in Tutorial-1 (Section 4.1.2). The tutor intervenes just for asking questions, making comments, drawing students' attention to relevant points, validating students' solutions, etc, but *not* for presenting solutions to the problem.

4.4 Monday-2: Problem-solving procedure

Under tutor guidance, students practice problem-solving procedures in a third 3-hour session (in addition to the two sessions of Wednesday-1). The activity aims at continuing practice of items of Wednesday-1 and at developing practice of:

- using Laplace transform for computing parameters of continuous random variables;
- using Generating transform for computing parameters of discrete random variables.

Exercises of the present activity have been carefully elaborated for practicing relevant knowledge to solve Items 5 to 7 and Item 9 of the solution alternatives (Section 4.1.2), in addition to continuing

the practice of knowledge related to Items 1-4 and 8 (started on Wednesday-1). Therefore, after this session, students should be able to solve the whole problem.

4.5 Tuesday-2: Problem-solving validation

In a 3-hour session, students validate their solutions in the presence of a supervisor (tutor or assistant). More precisely, students:

- explain to the supervisor the method used to solve each item of the solution alternatives (Section 4.1.2);
- present the results obtained. More precisely, the students generate their solutions for given values of *k, b, r, δ* and λ, and then, the supervisor checks correctness by comparing these solutions to his own (correct) solution.

The supervisor validates solutions, makes comments, draws students' attention on missing or incorrect points, but does *not* provide any correct method or result.

4.6 Wednesday-2: Tutorial-2

Each group of students has a second 90-minute tutorial meeting (denoted Tutorial-2). Under tutor guidance, students reflect on what they have learned, and determine if anything is missing in their understanding of the problem. By asking questions, the tutor helps students in the following steps:

Validation of knowledge acquired: *(60 minutes)* Students:

- review conclusions that were generated in Tutorial-1 (see Section 4.1.2), that is: a succinct formulation of the problem, and solution alternatives;
- state the concepts that have been used in their study. The tutor makes sure that all essential concepts are reviewed, and checks if necessary knowledge (see Sect. 3.2) is acquired correctly.
- generalize and de-contextualize the new knowledge. For example:
  - Probabilities: evaluating chances to win in gambling (lottery, casino, cards, …), signal processing (noise filtering), etc.
  - Laplace transform: transfer function linking a continuous input signal to a corresponding continuous output signal, used for example in automation and signal processing.
  - Generating function (or z-transform): like Laplace transform, but for *discrete* signals.

- Queuing systems: persons waiting for using a public phone, commands sent to a printer, batch processing system, etc.

- Integrals: computing surfaces and volumes, computing speed from acceleration, computing energy from power, etc.

Assessment of learning: *(30 minutes)* Students:

- report on knowledge acquired and on proposed solutions. They determine among necessary knowledge elements identified in Tutorial-1 (see list of Section 3.2), those that are operational and those that necessitate additional learning.
- discuss on their learning strategies.
- give their opinion about the learning and the atmosphere during the PBL unit.

Students also submit a written report presenting, in about 8 pages, what has been learned in solving the problem.

The remaining activities are related to assessment, and thus, are presented in the next section.

5. Assessment in APP_{prob}

Assessment is a very important issue in teaching and learning, which is confirmed by:

- the development of important resources related to assessment in many universities and institutes (e.g., Assessment Resource Centers (ARC) [6, 7], and the Assessment and Evaluation staff of the Teaching and Learning Laboratory (TLL) at MIT [8]);
- international conferences related to assessment [9, 10];
- assessment forums [11] and assessment websites [12];
- many publications related to assessment (e.g., [13-22]).

The assessment principles we have applied are mainly inspired from [23, 2]. Since APP_{prob} aims at developing the four competencies introduced in Section 3.1, assessment must be elaborated carefully in such a way as to allow accurate evaluation of these competencies. A *formative* written assessment, consisting of several problems, is provided to students (at the end of Wednesday-2) with a detailed model answer for each problem. Besides, competency(ies) involved in each question of a problem are identified, indicated and weighted. By *weighted*, we mean that a number is associated to the competency for measure purpose. Students can thus: check their individual learning achievement by comparing their answers to the model answer,

and measure to which level each competency is developed. And last but not least about formative assessment, students can evaluate their preparation to the written exams (see below). In order to determine if a student passes or fails the unit, (s)he will be evaluated through:

- the report submitted during Tutorial-2 (see Section 4.6), which presents clearly what has been learned in solving the problem. This is a final and written report of activities of Tuesday-2.

- two written exams, referred to as *summative* assessments, at the end of the unit (i.e., Friday-2) and at the end of the trimester, respectively.

Similarly to formative assessment, competency(ies) involved in each question in a summative assessment, are identified, indicated and weighted. So far, assessment has been distributed among the four competencies as follows: C_1: 25 % C_2: 18 % C_3: 17 % C_4: 40 %

To determine if a student passes the unit, let us consider the three possible situations:

- A student passes the unit if (s)he is evaluated at least 50 % for each of the four aimed competencies.

- A student fails the unit if (s)he is evaluated below 50 % globally (i.e., average over the four competencies is below 50 %). The student must then take the unit the next time it is provided.

- A student fails a competency if (s)he is evaluated below 50 % in this competency and at least 50 % globally. In order to pass the unit, the student must be assessed again and evaluated at least 50 % in each failed competency.

The only allowed document during summative assessments is a summary of probabilistic formulae.

Note that this PBL unit is worth 2 credits, with a total of 15 credits for the trimester, and a total of 120 credits for the whole computer engineering program.

6. Discussion

Let us compare the PBL approach to "conventional" courses in the context of APP_{prob}. The latter takes place during Trimester 3, and its main learning can be summarized by: *use of probabilities for analyzing computer systems*.

In our previous (course-based) computer engineering programs, students learned probabilities by acquiring related declarative knowledge (see Sect. 3.2.1), and then they applied some of the acquired

knowledge in solving very simple exercises. Thus, contextualization was not used at the early stage of learning. Moreover, the use of probabilities for solving non-trivial problems was programmed only during Trimester 8 in an optional course entitled *performance analysis* (PA) which has never been taught. Hence, many students terminated their bachelor's degree without having applied probabilities in solving non-trivial problems. We noted that when confronted to a problem necessitating the use of probabilities, many students had a preference to use probabilities in a very intuitive way, instead of using a rigorous approach. As a consequence, erroneous results were frequent. For example, for computing the average value of a discrete random variable (DRV), some students assumed subconsciously (and unduly) the DRV to be uniform. As for a continuous random variable (CRV), many students were unable to compute its average value even if the CRV is uniform. Many students preferred to use systematic methods for computing probabilistic parameters, without mastering the underlying concepts and theory. Those students were also unable, for example, to estimate the influence of certain parameters on the performance.

With the new programs, students have to solve *real* problems. In APP_{prob}, the problem is simple, but students need to think about and master many concepts related to the list of knowledge of Sect. 3.2.1. Another ability developed with the new programs is *integration*. In APP_{prob}, students integrate several concepts that were studied separately in conventional programs, such as probabilities, Laplace transforms, generating functions, queuing systems, and computer systems.

Therefore, with the new programs, students are better prepared for analyzing performance of computer systems. This is an important advantage, because students are well prepared for their first industrial training (beginning after Trimester 3). Furthermore, they will be better prepared in subsequent trimesters (i.e., 4-8), for improving their competencies and developing new competencies.

At the end of each PBL unit, a responsible of the trimester has a meeting with students in order to receive their comments about the unit. And at the end of the trimester, supervisors involved in PBL units or in the project, have a meeting with students in order to receive their global comments. Student feedback has been positive for the trimester; and in particular for APP_{prob} it has been very encouraging in many aspects, such as their learning and interest, and their appreciation of tutors and assistants.

One can say that with PBL approach, students learn how to learn.

7. Conclusion

The Department of Electrical and Computer Engineering of the *Université de Sherbrooke* has undertaken a major reform of its programs. The new pedagogical approach is based on *competence development* for *solving problems* and *realizing design projects*. As an illustration of the problem-based learning (PBL) approach, we present a two-week PBL unit that aims at developing and assessing competencies in applying probabilities in computer engineering. Student feedback has been very positive and encouraging in many aspects, such as their learning and interest, and their appreciation of tutors and assistants. This experience is encouraging especially since it shows that, contrary to certain prejudice, PBL is also applicable to theoretical subjects.

As a future work, it would be interesting to investigate how *Action Research* [24, 25] can contribute to the improvement of our supervision and assessment methods in PBL and competency development.

References

[1] Canadian Academy of Engineering. Evolution of Engineering Education in Canada, 1999. http://www.acad-eng-gen.ca/publis/publi_an.html.

[2] G. Lachiver, D. Dalle, N. Boutin, A. Clavet, F. Michaud, and J.-M. Dirand. Competency- and Project-Based Programs in Electrical & Computer Engineering at the Université de Sherbrooke. *IEEE Canadian Review*, pages 21—24, summer 2002.

 [3] A. Khoumsi and R. Gonzalez-Rubio. Applying a Competency- and Problem-Based Approach for Learning Compiler Design. Accepted in *the Journal of STEM Education: Innovations and Research*, 2005.

 [4] A.O. Allen. *Probability, Statistics, and Queuing Theory with Computer Science Applications*. Academic Press (2nd edition), 1990.

 [5] D. Woods. Problem-Based Learning: resources to gain the most from PBL, 1996. http://chemeng.mcmaster.ca/pbl/append-a.htm.

 [6] Assessment Resource Center of the University of San Francisco. http://www.usfca.edu/assessment/asses1.htm.

 [7] Assessment Resource Center of the University of Maryland University College (UMUC). http://www.umuc.edu/distance/odell/irahe/arc/.

 [8] Assessment and Evaluation staff of the Teaching and Learning Laboratory (TLL) at MIT. http://web.mit.edu/tll/assessment.htm.

[9] AAHE Assessment Conference series (since 1998). http://www.aahe.org/convenings.htm#assessment.

[10] Assessing Quality in Higher Education (AQHE) Conference series (since 1993). http://www.planning.iupui.edu/conferences/international/internationalconf.html.

[11] Assessment Forum of the American Association for Higher Education. http://www.aahe.org/initiatives/assessment.htm.

[12] Assessment Websites. http://www.aahe.org/assessment/web.htm.

[13] Y.J. Dori and J. Belcher. How Does Technology-Enabled Active Learning Affect Undergraduate Students' Understanding of Electromagnetism Concepts? *The Journal of the Learning Sciences,* 14(2), 2004.

[14] W. Ellis and M. Ratcliffe. An Anonymous Approach to Group Based Assessment. In *8th Computer Assisted Assessment Conference*, Loughborough, UK, July 2004.

[15] H. Levinson. *Organizational Assessment: A Step-by-Step Guide to Effective Consulting*. Washington, D.C.: American Psychological Association, 2002.

[16] J.E. Heady, B.P. Coppola, and L.C. Titterington. *Assessment Standards. College Pathways to the Science Education Standards*. Arlington, Va.: National Science Teachers Association Press, 2001.

[17] N.J. Entwistle. Approaches to Studying and Levels of Understanding: The Influence of Teaching and Assessment. *Higher Education: Handbook of Theory and Research*, 15:156—218, 1999.

[18] L. Suskie. Fair Assessment Practices: Giving Students Equitable Opportunities to Demonstrate Learning. *AAHE Bulleting*, 52(9):7—9, 2000.

[19] P.T. Ewell. National Trends in Assessing Student Learning. Journal *of Engineering Education*, 87(2):107—113, 1998.

[20] T.A. Angelo. Doing Assessment as if Learning Matters Most. *AAHE Bulleting*, 51(9):3—6, 1999.

[21] P. Black and D. William. Assessment and Classroom Learning. *Assessment in Education*, 5(1):7—74, 1998.

[22] V. Fabry, R. Eisenbach, and V.L. Gordon. Thanks for Asking: Classroom Assessment Techniques and Students' Perceptions of Learning. *Journal of Excellence in College Teaching*, 8(1), 1997.

[23] M.E. Huba and J.E. Freed. *Learner-Centered Assessment on College Campuses. Shifting the Focus from Teaching to Learning*. Boston: Allyn and Bacon, 2000.

[24] R.L. Baskerville. Investigating Information Systems With Action Research (Tutorial). *Communications of the Association for Information Systems*, 2(19), 1999. http://cis.gsu.edu/%7Erbaskerv/CAIS_2_19/CAIS_2_19.html.

[25] Action Research Tutorial. http://www.clay.k12.fl.us/roe/ActionResearchTutorial.htm.

Ahmed Khoumsi received the Engineer degree in Aeronautics and Automation from the engineer school SUP'AERO (Toulouse, France), in 1984. From 1984 to 1988, he achieved his research activities in the LAAS, a CNRS research center, in Toulouse. In 1988, he received the Ph.D. degree in Robotics and Automation from the University Paul Sabatier in Toulouse. From 1989 to 1992, he was an Assistant Professor in Robotics and Computer Engineering at the engineer school ENSEM (Casablanca, Morocco). From 1993 to 1996, he was a Postdoctoral Fellow in the Communication Protocols group of the University of Montreal. From 1996 to June 2000 he was an Assistant Professor, and since then he is an Associate Professor in the Department of Electrical and Computer Engineering, at the University of Sherbrooke, in Canada. His present research activities include: modeling, testing, and control of distributed and real-time systems; design and creation of telecommunications services; and application of new learning methods in teaching.

Brahim Hadjou received the Master degree in Mathematics from the Mathematical Institute of Constantine University (Algeria), in June 1986. He earned his Ph.D. degree in Mathematics from the University of Sherbrooke (Canada), in December 1992. From 1989 to 2001, he taught mathematics in both Faculties of engineering and of science at the University of Sherbrooke, and since then he is an Assistant Professor in the Department of Electrical and Computer Engineering, at the same University. His research interests are applied mathematics in engineering and application of new learning methods in teaching.