A Model for Designing Learning Experiences for Computer Science Curriculum

Ioana Ghergulescu, Paul Stynes & Pramod Pathak*
School of Computing
National College of Ireland
Mayor Street, IFSC, Dublin 1, Ireland
ioana.ghergulescu@ncirl.ie; paul.stynes@ncirl.ie; pramod.pathak@ncirl.ie
*Corresponding Author

***Abstract*—Due to the fast development in computer science new modules and specializations have to be developed, and the Computer Science (CS) curriculum needs to be reshaped in order to include 21st century skills such as problem-solving, creativity, innovation, communication and collaboration. This paper proposes a generic model for designing learning experiences for CS curriculum. The model builds on existing credit systems such as European Credit Transfer and Accumulation System (ECTS), to determine if a CS programme can be reshaped to a given period of time. Furthermore the model is capable of determining the student workload distribution across the different types of learning activities for a module. The distribution is computed based on information such as the number of credits for the module, the number of weeks in a semester, as well as the percentage hours of lectures, labs, independent study and other learning activities. Preliminary data collection and analysis was conducted in order to determine the percentages on 75 computer science modules taught at 14 universities from Ireland and UK.**

Keywords—computer science; pedagogy, curriculum design;

I. INTRODUCTION

With the fast technological developments in the computer science area, there is increasing demand for the provision of new modules and specialisations. However, updating computing curriculum or designing new curriculum is a challenging task. This includes the elaboration of the learning "journey" from several aspects: aims, expected learning outcomes, learning experiences (i.e., learning and teaching methods and activities), and assessment. However, pedagogies for 21st century learners are still to be fully validated. According to industry feedback the gap is in the area of problem-solving ability, creativity, innovation, communication and collaboration: these are essential 21st century skills [1]. Current trends in computer science education have not addressed this gap significantly. There is a need for providing models and examples of how the 21st century skills can be related to the subject domain as well as for providing help to restructure the curriculum [2].

This work in progress addresses the challenge of designing and structuring the learning activities for computer science modules that incorporate 21st century skills. The goal is to provide a model on how to structure the modules in terms of the learning activity type (i.e., lectures, laboratories/tutorials, individual learning, and other learning activities), based on the number of credits per module and the study/semester duration. The model also takes into account the advances in computer science pedagogy in order to maximize students' retention, learning outcomes, motivation and employability. The research question that this work in progress addresses is: What is the recommended structure and duration of a computer science module for a semester?

The model is a generic one and can be used to reshape or design any computer science module. Therefore this model is of value to those involved in curriculum design such as educators and programme directors. The model builds on recommendations from the three most used standard credit systems (ECTS, Framework for Higher Education Qualifications (FHEQ) and U.S. Course and Credit System). The model can be utilised for reviewing the programmes as well as creating new programmes. In its current form, the model is able to detect if a computer science programme of study can be reshaped to a given period of time, as well as to provide the distribution of learning activities.

The remainder of this paper is structured as follows. Section 2 presents the background and previous research works on credit systems and pedagogy for 21st century skills. Section 3 presents the proposed innovative model for designing learning experiences for CS curriculum. Section 4 details the data analysis to detect the workload percentages for the different learning activities. The final section concludes the paper and indicates future research directions.

II. RELATED WORKS

A. Credit Systems

The *credit* represents a unit that gives weight to the value, level of an academic course taken at an educational institution. *Notional time* represents the number of hours expected that a learner (at a particular level) will spend, on average, to achieve the specified learning outcomes (through all Learning Activities) at that level. *Student workload* represents the amount of work or working time expected or assigned within a specific period of time for a specific number of credits.

There are several different credit systems used in higher education, including: The European Credit Transfer and Accumulation System (ECTS); The Framework for Higher Education Qualifications in England, Wales and Northern Ireland (FHEQ), and U.S. Course and Credit System [3].

ECTS is a "learner-centred system for credit accumulation and transfer based on the transparency of learning outcomes and learning processes" [4]. ECTS is used for formulating qualification frameworks for higher education by 46 countries engaged in the Bologna Process. In the ECTS system each academic year on a regular bachelor programme of study is

978-1-4799-3922-0/14/$31.00 ©2014 IEEE

allocated a total of 60 credits (i.e., 30 credits per semester, 240 credits per 4 year bachelor course). ECTS assumes a student workload of 1500 to 1800 hours per academic year, which correspond to 25-30 notional hours per 1 credit. However, ECTS does not standardises the semester duration, and thus the expected workload varies across countries and universities. In Ireland, the National Framework of Qualifications recommends 20 to 30 notional hours per ECTS credit [5], while for example University College Dublin expects 20 to 25 hours of total student effort for each credit [6].

The FHEQ credit system specifies that bachelor degrees with honours in England, Wales and Northern Ireland, require typically a total volume of at least 360 credits, which equate to 180 ECTS credits as long as the learning outcomes are consistent [7]. FHEQ recommends a number of 10 notional hours for each UK credit (i.e., 20 hours per ECTS credit).

In the U.S. Course and Credit System, a typical bachelor's degree program of study on a semester calendar requires at least 120 credit hours (i.e., Carnegie units) to be earned by the student [3]. Each academic year has allocated 30 credit hours (i.e., US credit), with 0.5 credit hours being equivalent to 1 ECTS credits, and a number of 26 to 28 notional hours [8].

B. *Computer Science Pedagogy for 21st Century Skills*

Previous research in computer science pedagogy suggests that both the lectures and the practical activities are very important for computer science education [9]. The literature shows a trend towards the introduction of various learning activities such as: hands-on programming challenges; collaborative in-course problem solving; peer instruction; and pedagogical code reviews, etc.

Furthermore, the literature shows that students in an active learning environment demonstrate a higher rate of retention and mastery of the material and lower dropout rates [10]. In order to enhance the student's learning outcomes and motivation, the authors from [11] have proposed to combine theory in computer science with hands-on programming challenges and collaborative in-course problem solving. The main features of their pedagogical approach include collaborative learning, constant activities to stimulate interactions between instructor and students, immediate feedback for self-assessment, and motivational feedback such as badges or leaderboard information. In [12] the authors proposed to introduce peer instruction and pedagogical code reviews into the computer science classroom.

When rethinking pedagogy for a digital age, the educators should design the learning activities in a way that helps the students gain 21st century skills [13]. The 21st Century Skills can be divided into three main categories [14]:

- Learning and innovation skills (critical thinking, communication, collaboration, reactivity).
- Information, media and technology skills (information literacy, media literacy, ICT literacy.
- Life and career skills (flexibility and adaptability, initiative and self-ddirection, social and cross-cultural skills, productivity and accountability, leadership and responsibility.

In order to help the students gain these skills several activities were proposed such as: social activities (discussions should be added to lecturers, reading and practice activities); digital literacy development (using technology for learning that may include blogging, webinars); 21st century learning activities (seminars on essential skills for 21st century); real life projects, etc. [15].

The authors in [16] have proposed a model for teaching and learning where the key element is represented by a team-based pedagogy. In this model, the learners have access to a physical learning space to support the team-based learning and the teachers are guiders and mentors. Moreover, the students lead the projects; technology is used as an integral tool in the process, while team and individual reflection is incorporated as part of a cross-curriculum thematic learning.

However, the research showed that 21st century skills are not well implemented in the educational practice as the skills implementation requires restructuring the curriculum [2].

III. Model Description

This section presents the proposed model for designing learning experiences for CS curriculum. The model aims to detect if a computer science programme of study can be reshaped (or creation of a new programme) to a given period of time with a recommendation of weekly student workload distributed across the different types of learning activities for a module. To achieve this, the model takes into consideration the recommendations from the three most used standard credit systems (ECTS, FHEQ, U.S. Course and Credit System), and makes use of information such as the number of credits for the module, the number of weeks in a semester, as well as the percentage hours of lectures, labs, independent study, specific learning activities and other learning activities. Furthermore, the model takes into consideration the module categorisation with three categories, namely: theoretical computer science modules, applied computer science modules, and business modules for computer science.

A. *Overview*

The first two steps employed by the proposed model for designing learning experiences for CS curriculum are: i) student workload limits validation, and ii) computation of workload hours for the different types of learning activities.

1) Student Workload Limits Validation

The number of credits over a given period of time T for a module M *is denoted by* $NC(T)$. For example $NC(Year) = 60$ ECTS credits. The student workload for a given number of credits in a period of time T, W(T) is expressed in (1).

$$W(T) = NT * NC(T) \quad (1)$$

where, W represents the student workload, NT represents the notional time, and NC represents the number of credits.

For example, in case of ECTS, the student workload per semester can range between 600 and 900 hours, when considering a workload of 20 to 30 notional hours for each of the 30 credit points allocated to a semester.

Based on the semester workload, the daily workload can be computed as in (2). In the equation a semester includes both the actual teaching weeks and additional weeks when students are normally expected to study such as reading and examination weeks, as expressed in (3).

$$DW(Semster) = W(Semester) / (NWS * NDW) \quad (2)$$

$$NWS = NTW + NEW \quad (3)$$

where, *DW* represents the daily workload, *NWS* represents the number of weeks per semester, *NDW* represents the number of study days per week, *NTW* represents the number of teaching weeks, and *NEW* represents the number of extra weeks.

For example, considering *NWS* = 12 + 5 =16 and *NDW* = 5, it results that: *DW* = [600 / (16 * 5), 900 / (16 * 5)] = [7.5, 11.25] hours.

2) Computation of the Distribution of Learning Activities

The student workload for a particular module with a particular number of credits is expressed in (4).

$$W(Module) = NT * NC(Module) \quad (4)$$

For example, in case of ECTS, and considering the "Networking Programming and Distributed Systems (NPDS)" module, the computed workload would be between 100 and 150 hours. This corresponds to a workload of 20 to 30 notional hours for each of the 5 credits allocated to the NPDS module. The workload for a particular module can be further expressed in (5).

$$W(Module) = CLecH(Module) + CLabH(Module) + ISH(Module) + SLAH(Module) + OLAH(Module) \quad (5)$$

where, *CLecH*, *ClabH*, *ISH*, *SLAH* and *OLAH* represent the number of hours for the contact lectures, contact lab, individual study, specific learning activities, and other learning activities respectively.

B. Model Algoritms

Algorithm 1 presents the procedure for detecting if a programme of study can be scheduled in a given period of time (e.g., 12 weeks of teaching per semester and 4 weeks of additional study). This has to be done by taking into account the number of credits for the programme of study, the number of weeks in a semester, as well as the maximum reasonable number of student workload hours per day. The student daily workload has to be computed and compared against the specified maximum threshold. If the computed daily workload is smaller than the specified maximum reasonable number of daily workload hours, then it is feasible to schedule the module to the given duration of time.

Algorithm 2 presents the procedure used to determine the distribution of learning activities for the particular module in a given period of time. This is done by taking into account the number of credits for the particular module and the number of weeks in a semester, as well as the module category and the percentages of lectures, labs, independent study, specific and other learning activities for that category. The percentages can be computed based on data collected from Ireland and UK universities.

Based on the percentages and the semester workload the total numbers of hours for lectures, labs, independent study, specific and other learning activities in a semester, are computed. Based on these, the number of hours for lectures, labs, specific and other learning activities in a teaching week, are computed. If the computed number of weekly hours are not rounded, these will have to be compensated. Strategies that can be used include: increase/decrease the number of weekly contact hours, unequally distribute the number of contact hours over the semester duration, and/or replacing some contact hours with independent/other learning activities.

Algorithm 1: Detection of the possibility to schedule a program with a specific number of credits in a specific amount of time.

INPUT
NC – number of credits for the programme of study;
NWS – number of weeks per semester;
ThMaxDW – maximum reasonable number of workload hours per day.
PROCEDURE
begin
  Compute *W*(semester) for the particular programme
  Compute *DW*
  if *DW* < *ThMaxDW* **then** *IsPossible* = TRUE
  else *IsPossible* = FALSE
end
OUTPUT
IsPossible – flag indicating the possibility to schedule the programme.

Algorithm 2: Detection of the distribution of learning activities for a particular module.

INPUT
NC – number of credits for the module;
NTW – number of teaching weeks from a semester;
C – module category
PCLecH, PCLabH, PISH, PSLAH, POLAH – sets of percentages of lectures, labs, independent study, specific learning activities and other activities.
PROCEDURE
begin
  Compute *W* (module)
  Compute total number of hours per semester:
  *CLecH = PCLecH[C] * W* (module)
  *CLabH = PCLabH[C] * W* (module)
  *ISH = PISH[C] * W* (module)
  *SLAH = PSAH[C] * W*(module)
  OLAH = POLAH [C] W(*module)
  Compute number hours per week:
  CLecHW = CLecH /NTW
  CLabHW = CLabH /NTW
  SLAHW = SLAH /NTW
  OLAHW= OLAH/NTW
  if (compensationNecessary) **then**
    Compensate (*CLecHW, CLabHW, SLAHW, OLAHW*)
end
OUTPUT
CLecHW, CLabHW, SLAHW, OLAHW - represent the number of hours per week for the contact lectures, contact lab, specific learning activities, and other learning activities respectively

IV. Learning Activities Percentages Detection

A. Data Collection

Data on 75 computer science modules taught at 14 universities from Ireland and UK, was collected and used in the analysis. The data collected on each module includes among others: the number of credits, the learning activity types (lecture, lab/tutorial, independent study, other activities, etc.), and the number of hours per learning activity.

B. Results

The preliminary data analysis was conducted in order to investigate the number of hours for the five main types of learning activities (i.e., lectures, lab/tutorial, independent study, specific and other activities). To make a fair comparison, the number of hours for each module was normalised relative to the number of credits for that module,

and multiplied by 5 (i.e., the number of credits of a standard computer science module). The data was collected from the websites of the universities. The averages and the percentages were computed by taking into account all the available data as not all of them provided data for all the different learning activity types. Table I presents the average and standard deviations for workload hours for the different learning activity types across the universities form Ireland and UK.

TABLE I. AVERAGE NUMBER OF HOURS FOR THE DIFFERENT LEARNING ACTIVITY TYPES IN IRELAND AND UK

Learning Activity	Ireland		UK	
	AVG	*STD*	*AVG*	*STD*
Lectures	22.36	4.98	15.07	5.03
Laboratory	20.89	13.58	12.91	5.36
Independent Study	56.96	17.14	70.11	8.15
Other Learning Activities	0.94	4.02	0.70	1.23
Specific Learning Activities	7.32	16.81	0.80	4.00
Total Hours	111.39	13.14	99.60	1.38

Table II presents the workload hours as average percentages for each learning activity types in case of the three categories of CS modules taking into consideration data from both Ireland and UK. The percentages of learning activities were computed for each individual module and then averaged across all the modules from a category.

TABLE II. AVERAGE PERCENTAGES OF LEARNING ACTIVITIES FOR THE THREE CATEGORIES OF CS MODULES

Learning Activity	CS Modules Category		
	Business for CS	*Theoretical CS*	*Applied CS*
Lectures	20.92%	20.02%	16.00%
Laboratory	13.03%	17.81%	17.81%
Independent Study	63.44%	55.12%	60.15%
Other Learning Activities	0.00%	1.12%	0.86%
Specific Learning Activities	2.61%	5.93%	5.18%

For example, the NPDS module that is an applied CS module, will have 16.00% lectures, 17.81% laboratory, 5.18% specific learning activities, 0.86% other learning activities. For a 12 weeks teaching duration in a semester and a workload in the [100, 150] range, the number of lecture hours should be in the [16, 24] range, the number of laboratory hours should be in the [18, 27] range, specific learning activities should be in the [5, 8] range, and 1 hour should be allocated for other learning activities. Examples of other learning activities include: video material, clinic session, peer-programming, code review, collaborative learning activities, etc.

V. CONCLUSION AND FUTURE WORK

Due to fast developments in computer science new modules and specialisations are being developed, and the CS curriculum needs to be reshaped in order to incorporate 21st century skills. Addressing these challenges, this paper has proposed a generic model for designing learning experiences for CS curriculum. The model builds on the standard CS curriculum and the existing credit systems such as ECTS. As the literature review has shown a trend towards integrating specific learning activities in the curriculum, the model takes into consideration not only lectures and labs but also other specific learning activities. The model first detects if a particular module can be reshaped and then detects the percentages of workload hours for each learning activity type.

Specific learning activities should be created in order to promote higher order thinking skills in our students. The students should not only be able to remember, understand and apply but also to analyse (compare, organize questions, research, deconstruct, outline, attribute), to evaluate (check, judge, critique, experiment hypothesis, test, detect), and to create (design, build, construct, plan, practice, devise) [17].

Future work will focus on evidence-based pedagogical research in order to determine best types of learning activities for specific module categories. Moreover, the model will be extended by considering additional data from more universities in order to compute the percentage hours of learning activities. This will also explain the variation in learning activities for "CS modules categories". Furthermore, the model will be extended to consider other courses categorisations such as for example: the 18 Knowledge Areas from the ACM/IEEE-CS Computer Science Curricula 2013 and computer science pedagogy. The model is proposed to be used in programmatic review in National College of Ireland. Future work will also address the evaluation and refinement of the model.

REFERENCES

[1] Fortas, "Addressing Future Demand for High-Level ICT Skills," Fortas, 2013.

[2] J. Voogt, O. Erstad, C. Dede, and P. Mishra, "Challenges to learning and schooling in the digital networked world of the 21st century," *J. Comput. Assist. Learn.*, vol. 29, no. 5, pp. 403–413, Oct. 2013.

[3] USNEI, "Structure of the U.S. Education System: Credit Systems," International Affairs Office, U.S. Department of Education, 2008.

[4] European Communities, "ECTS Users' Guide," European Commission Education and Training, Brussels, Belgium, Feb. 2009.

[5] NQAI, "Principles and operational guidelines for the implementation of a national approach to credit in Irish higher education and training"," National Qualifications Authority of Ireland, Dublin, Ireland, Jul. 2006.

[6] UCD, "User's Guide to the General Regulations: 2013/2014 Academic Session, Modules and Credits," University College Dublin, Dublin, Ireland, 2014.

[7] QAA, "Higher education credit framework for England: guidance on academic credit arrangements in higher education in England," Quality Assurance Agency for Higher Education, Mansfield, UK, Aug. 2008.

[8] ISEP, "Transcript Evaluation Guidelines 2013," International Student Exchange Programs, 2013.

[9] A. Robins, J. Rountree, and N. Rountree, "Learning and Teaching Programming: A Review and Discussion," *Comput. Sci. Educ.*, vol. 13, no. 2, pp. 137–172, 2003.

[10] T. Briggs, "Techniques for Active Learning in CS Courses," *J Comput Sci Coll*, vol. 21, no. 2, pp. 156–165, Dec. 2005.

[11] J. Pirker and C. Gütl, "Motivational Active Learning for Computer Science Education (Abstract Only)," in *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, New York, NY, USA, 2014, pp. 711–711.

[12] S. Grissom, C. Hundhausen, and P. Conrad, "Alternatives to Lecture: Experience Peer Instruction and Pedagogical Code Reviews," in *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, New York, NY, USA, 2014, pp. 275–276.

[13] H. Beetham and R. Sharpe, *Rethinking Pedagogy for a Digital Age: Designing for 21st Century Learning*. Routledge, 2013.

[14] P21, "P21 Framework Definitions," The Partnership for 21st Century Skills, Washington, DC, USA, 2014.

[15] A. Samineni, S. Murali Mohan, and B. Mohan Murari, "A practical approach of revitalizing K-12 education in 21st century," in *2013 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)*, 2013, pp. 136–140.

[16] C. Conneely, J. Lawlor, and B. Tangney, "Technology, Teamwork and 21st Century Skills in the Irish Classroom," in *Shaping our Future: How the lessons of the past can shape educational transformation*, Marshall, K., Ed. Dublin, Ireland: Liffey Press, 2014.

[17] L. W. Anderson, D. R. Krathwohl, P. W. Airasian, K. A. Cruikshank, R. E. Mayer, P. R. Pintrich, J. Raths, and M. C. Wittrock, *A Taxonomy for Learning, Teaching, and Assessing: A revision of Bloom's Taxonomy of Educational Objectives*. New York: Pearson, Allyn & Bacon, 2000.