Developing a Computer Science-specific Learning Taxonomy

Ursula Fuller
Computing Laboratory
University of Kent
Canterbury CT2 7NF
United Kingdom
U.D.Fuller@kent.ac.uk

Colin G. Johnson
Computing Laboratory
University of Kent
Canterbury CT2 7NF
United Kingdom
C.G.Johnson@kent.ac.uk

Tuukka Ahoniemi
Institute of Software Systems
Tampere University of Technology
Tampere, Finland
tuukka.ahoniemi@tut.fi

Diana Cukierman
School of Computing Science
Simon Fraser University
Burnaby, British Columbia
Canada
diana@cs.sfu.ca

Isidoro Hernán-Losada
Lenguajes y Sistemas Informáticos
Universidad Rey Juan Carlos
Madrid
Spain
Isidoro.hernan@urjc.es

Jana Jackova
Faculty of Management Science and Informatics
University of Zilina /Slovak University of Technology
Zilina, Slovak Republic
Jana.Jackova@fri.uniza.sk

Essi Lahtinen
Institute of Software Systems
Tampere University of Technology
Tampere
Finland
essi.lahtinen@tut.fi

Tracy L. Lewis
Information Technology
Radford University
Radford, VA 24142
USA
Tlewis32@radford.edu

Donna McGee Thompson
Student Learning Commons
Simon Fraser University
Burnaby, British Columbia
Canada
dmcthomp@sfu.ca

Charles Riedesel
Computer Science & Engineering
University of Nebraska Lincoln
259 Avery Hall
Lincoln, Nebraska 68588-0115
USA
riedesel@cse.unl.edu

Errol Thompson
Massey University
Wellington
New Zealand
kiwiet@computer.org

ABSTRACT

Bloom's taxonomy of the cognitive domain and the SOLO taxonomy are being increasingly widely used in the design and assessment of courses, but there are some drawbacks to their use in computer science. This paper reviews the literature on educational taxonomies and their use in computer science education, identifies some of the problems that arise, proposes a new taxonomy and discusses how this can be used in application-oriented courses such as programming.

Keywords

Computer science education, taxonomies of learning, curricula, assessment, credit transfer, benchmarking

Categories and Subject Descriptors

K.3.2 Computer and Information Science Education

General Terms

None

1. INTRODUCTION

1.1 Motivation

Educational taxonomies are a useful tool in developing learning objectives and assessing student attainment. They can also be deployed in educational research, for example to classify test items and investigate the range of learning these are measuring. The well-known educational taxonomies are generic and rely on the assumption that the hierarchy of learning outcomes is the

same in all subjects, from art history to zoology. However, taxonomies are not simple to use and researchers find it hard to reach agreement on the classification of items, which limits their benefits to instructors [27]. This paper reports the work of an ITiCSE Working Group investigating the hypothesis that the hierarchy of learning outcomes in computer science is not well captured by existing generic taxonomies and that computer science education would be better served by the development of a computer science-specific taxonomy.

1.2 What is an educational taxonomy?

A taxonomy is a classification system that is ordered in some way. Linnaeus's taxonomy arranged living organisms into a tree-structured hierarchy. This gave biologists a tool to help them understand the relationship between members of the plant and animal kingdoms and to communicate accurately about them [7]. Taxonomies of educational objectives can similarly be used to provide a shared language for describing learning outcomes and performance in assessments. Unlike the biological taxonomy, educational taxonomies are not usually tree-structured. To a greater or lesser extent they divide educational objectives into three domains, *cognitive, affective* and *psychomotor.* Some, such as Bloom's taxonomy, treat each of these as a one-dimensional continuum [7], others, like the revised Bloom's taxonomy, describe the cognitive domain using a matrix [3]. Yet others, like the SOLO taxonomy, use a set of categories that describe a mixture of quantitative and qualitative differences between the performance of students [5] and there are also taxonomies that claim they can be applied equally to all three domains.

1.3 What taxonomies are used for

Learning taxonomies describe and categorize the stages in cognitive, affective and other dimensions that an individual may be at as part of a learning process. Paraphrasing Biggs [6], we can say that they help with "understanding about understanding" and "communicating about understanding". Thus learning taxonomies can be seen as a *language* which can be used in a variety of educational contexts.

Learning taxonomies can be used to define the curriculum objectives of a course, so that it is not only described on the basis of the topics to be covered, but also in terms of the desired level of understanding for each topic [48]. Computing programs accredited by ABET have to be specified in terms of measurable objectives, including expected outcomes for graduates [14]. More generally, the use of learning outcomes is mandated in the countries of the European Higher Education Area [1,8,68] and is increasingly prevalent in the US and elsewhere [15].

Learning taxonomies are widely used to describe the learning stages at which a learner is operating for a certain topic. For example, a student may be capable of reciting by heart what recursion is but not capable of implementing a recursive algorithm. An instructor may aim to have his or her students learn a topic at a certain level in a taxonomy (e.g. students may be expected to be able to comprehend the concept of recursion without necessarily applying it). Once this has been done, the instructor can assess students at the chosen level through a suitable choice of questions or examples [39]. This approach is encouraged by teacher-trainers [26]. Furthermore, the students' answers can be analyzed as belonging to one level or another; such answers can help the instructor revise his or her teaching techniques to better guide students to accomplish a certain learning stage.

Learning taxonomies have been used in many other contexts, such as introducing students to a learning taxonomy to raise their awareness and improve their level of understanding and their studying techniques [16,71]. They are also used to structure exercises in computer-based and computer-assisted instruction [21,36].

1.4 Weaknesses of taxonomies from a CS standpoint

Learning taxonomies, particularly Bloom's taxonomy of the cognitive domain, have had a considerable impact on curriculum and assessment design in the last fifty years. However, this does not mean that their use is unproblematic. The classification of a specific learning outcome or test item depends on its context. A task that challenges the analysis and synthesis skills of a beginner becomes routine application of knowledge for a more advanced learner. Similarly, a student who has been taught how to solve a problem that is extremely similar to the test item will demonstrate skills lower in the taxonomic order than one who is solving it from first principles. This is a generic problem but computer science-specific difficulties also manifest themselves.

Johnson and Fuller [27] found that colleagues disagreed about the relative difficulty of cognitive tasks in computer science. A significant proportion felt that it is easier to apply knowledge to solve simple problems than to describe this knowledge. They also found that computer science instructors did not find the terms *synthesis* and *evaluation* useful in describing learning outcomes and assessment tasks for programming courses, especially at the introductory level, instead seeing the application of knowledge as the highest skill that they should be developing. Close questioning revealed that application, as used by these colleagues, did in fact subsume analysis, synthesis and evaluation, leading Johnson and Fuller to propose a revised taxonomy with *higher application* as the highest level.

Lahtinen's recent work [37] shows that the ordering of cognitive tasks in Bloom's taxonomy is a very poor fit for the learning trajectories of some students tackling programming for the first time. In addition, the use of taxonomies is concentrated on the cognitive domain, even though learning in the affective domain is also essential for the formation of computer science practitioners. These problems led the working group to investigate whether a subject specific taxonomy would be of more use to computer science instructors than the existing generic ones.

1.5 Methodology

In order to investigate this hypothesis, our working group has reviewed a number of taxonomies described in the educational literature, together with the range of uses to which they are put. We have also reviewed studies in the computer science education research literature that use one or more taxonomies as an analytic tool. In addition we have looked at the practice of assessment in computer science both for novice programming and in two other typical subject areas, drawing on the experience of members of the working group and their colleagues. We have used this evidence to propose a new, computer science-specific taxonomy and to make recommendations about how it might be used. We concentrated on the cognitive domain because that is

the area in which there is existing research on the use of taxonomies in computer science.

2. REVIEW OF EXISTING TAXONOMIES

Educational researchers have developed a range of taxonomies, developmental stages and instructional design strategies aimed at helping educators develop learning outcomes, educational resources, and assessments. These taxonomies have been based on a range of educational theories and research. Readers interested in the theoretical foundations for the taxonomies reviewed by the working group should direct their attention to the referenced papers.

2.1 Cognitive domain

2.1.1 Bloom and revision

Of these taxonomies, the most widely cited in the literature reviewed by the working group is the original Bloom's taxonomy [7]. Bloom's taxonomy has six categories, where each category builds on the lower ones:

	1. Knowledge
Abilities and skills	2. Comprehension
	3. Application
	4. Analysis
	5. Synthesis
	6. Evaluation

Bloom's taxonomy has since been revised by Anderson et al [3]. The authors changed the nouns listed in the Bloom's model into verbs, to correspond with the ways learning objectives are typically described.

Categories	Cognitive processes
1. Remember	Recognizing, Recalling
2. Understand	Interpreting, Exemplifying, Classifying, Summarizing, Inferring, Comparing, Explaining
3. Apply	Executing, Implementing
4. Analyze	Differentiating, Organizing, Attributing
5. Evaluate	Checking, Critiquing
6. Create	Generating, Planning, Producing

Revised Bloom's taxonomy [3]

These taxonomies do not define a sequence of instruction but define levels of performance that might be expected for any given content element. A learner performing at a higher level is expected to be able to perform at the lower levels in the cognitive hierarchy. This could be interpreted as implying a sequential learning process. However, the taxonomy doesn't rule out the use of an iterative approach to learning the content.

The authors of the revised taxonomy acknowledge that there is a possible overlap in terms of the cognitive complexity among the higher level categories of the hierarchy. However, the midpoint of each of the higher level categories is seen as being more complex than the lower category [32,67]. For example, the cognitive process of Explaining in the Understand category may require a higher cognitive load than Executing in the Apply category in some contexts.

A key difference between the revised taxonomy and the original taxonomy is that the type of knowledge elements is also defined: A. Factual knowledge, B. Conceptual knowledge, C. Procedural knowledge, D. Metacognitive knowledge. This provides a matrix into which learning objectives are mapped.

		Separate Knowledge dimension			
		A Factual	B Conceptual	C Procedural	D Metacognitive
Cognitive process dimension	1. Remember				
	2. Understand				
	3. Apply				
	4. Analyze				
	5. Evaluate				
	6. Create				

Revised Bloom's Taxonomy [67]

2.1.2 Niemierko , Tollingerova, Bespalko

Other taxonomies of learning objectives have extended Bloom's taxonomy.

Niemierko and others claim that the three highest Bloom categories (higher thinking processes) cannot be ordered hierarchically in science subjects [50]. This has been used for the development of curricula in e.g. Slovakia, the Czech Republic and Poland [35]. He developed the "ABC" taxonomy of learning objectives [50] that are organized in two dimensions:

Levels	Categories of learning objectives
I. Knowledge	A. Remembering of knowledge B. Understanding of knowledge
II. Abilities and skills	C. Application of knowledge in typical problem situations D. Application of knowledge in unfamiliar problem situations

Niemierko's "ABC" taxonomy of learning objectives [50]

Applications in category D include the analysis, synthesis, and evaluation categories of Bloom's taxonomy.

Tollingerova's taxonomy [35] has five, hierarchically-ordered operation categories : 1. memory reproduction of knowledge, 2. easy thought operations with knowledge, 3. difficult thought operations with knowledge, 4. communication of knowledge, 5. creative thinking

According to Bespalko learning objectives can be expressed in two stages of abstraction and four activity levels [35, 50]:

I. Reproductive activities: 1. recognition (identification), 2. reproduction

II. Productive activities: 3. application, 4. creativity (transformation)

2.1.3 Critical thinking

Some researchers see Bloom's taxonomy as not giving enough emphasis to aspects of critical thinking. Critical thinking goes beyond the cognitive categories of the original Bloom's taxonomy to incorporate attributes of reflective judgment with respect to the value of what is being learned and to make

judgments on the reliability and authority of the associated knowledge.

The reflective judgement taxonomy by King and Kitchener has a total of seven stages that fall into three groups indicative of *pre-reflective* thought (Stages 1-3), *quasireflective* thought (Stages 4 and 5), and *reflective* thought (Stages 6 and 7) [29].

Facione's critical thinking taxonomy is more closely aligned with Bloom's taxonomy. It lists six critical thinking skills with appropriate sub-skills. To become a good critical thinker exhibiting self regulation, the person must engage in interpretation, analysis, evaluation, inference, explanation, and meta-cognitive self-regulation [18]. The revised Bloom's taxonomy [3] endeavours to capture some of these skills through the use of the knowledge dimensions and the inclusion of the meta-cognitive knowledge.

2.2 Unified domain taxonomy

There have been a number of attempts to produce a taxonomy that covers the cognitive (C), affective (A) and psychomotor (P) domains. The work of De Block [35] is an example of this approach:

1 Knowledge
   C: *Repeat, define, show, name, etc.*
   A: *Listen to opinion of others, accept notes, realize, etc.*
   P: *Show, imitate, understand sound, smell, taste, etc.*
2 Understanding
   C: *Describe, characterize, say in own words, explain, compare, etc.*
   A: *Accept opinions of others, answer questions, react to rules correctly, ask relevant questions, participate, etc.*
   P: *Demonstrate a principle, put together and disassemble something that is known, etc.*
3 Application
   C: *Solve, calculate, number, translate, illustrate, analyze, make, etc.*
   A: *React to rules automatically, accept norms and values, cooperate in a group, apply norms and rules, etc.*
   P: *Make, produce, try, repair, adapt, cook, cut, put together and disassemble something that is new, etc.*
4 Integration
   C: *Design, create, summarize, judge, decide, plan, etc.*
   A: *React to rules spontaneously, apply norms spontaneously and behave under rules, initiate cooperation, find satisfaction in behavior and work under society's rules, etc.*
   P: *Perform an activity fluently, without hesitation, without mistakes, automatically; work precisely, quickly, etc.*

Niemierko [50] describes the possibility of synthesizing an overall educational taxonomy.

2.3 Structure of the Observed Learning Outcome (SOLO)

The SOLO taxonomy makes no reference to cognitive characteristics of the learner's performance or to the affective dimension. It focuses on the content of the learner's response to what is being assessed. It endeavours to identify the nature of that content and the structural relationships within that content. The content could be designed to assess knowledge, cognitive skills, or underlying values. The taxonomy can be used to establish the relationships expected between these different types of content. It is left up to the assessor or course designer to define the type of content expected.

The SOLO levels are:

- Prestructural – not related to topic – disjoint – missed the point
- Unistructural – simple meaning, naming, focussing on one issue in a complex case
- Multistructural – 'shopping list' – disorganised collection of items
- Relational – understanding, using a concept that integrates a collection of data, understanding how to apply the concept to a familiar data set or to a problem
- Extended abstract – relating to existing principle, so that unseen problems can be handled, going beyond existing principles [5,6].

In defining these categories, Biggs and Collis [5] use three crucial characteristics. These are:

1. capacity – how many things are handled in the content – "a quantitative increase in what is grasped" [6]
2. relating operation – the way in which the content is related to the intended purpose – the integration of the components within the content
3. consistency and closure – the drawing of conclusions or bringing to closure that is consistent

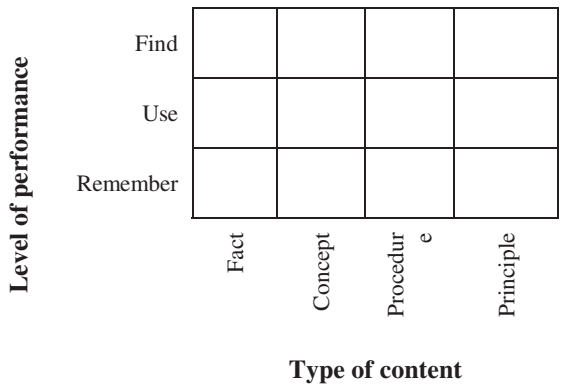
Using SOLO in assessment can provide a mechanism for holistic marking [69,70]. However, Biggs [6] provides examples of assessment strategies that use items targeted at specific SOLO levels as well as more holistic strategies.

The lower levels of the SOLO taxonomy (unistructural and multistructural) can be used to focus on individual items or attributes of what is being assessed. The higher levels with their emphasis on integration and extension of principles require a broader range of content or attributes to be examined.

The SOLO taxonomy makes no attempt to infer a cognitive processing level although it might be argued that to perform at a relational level or an extended abstract level involves greater cognitive processing than that required for unistructural or multistructural since the learners not only have to be able to recall items, they have to show the relationship among items (relational) and draw conclusions (extended abstract).

2.4 Instructional Design

Instructional designers use taxonomy concepts to guide course creation. Merrill proposed the Component Display Theory (CDT) for instructional design [46,47]. It classifies learning along two dimensions: content (facts, concepts, procedures, and principles) and performance (remembering, using, and generalizing). A complete lesson would consist of an objective followed by some combination of rules, examples, recall, practice, feedback, helps and mnemonics appropriate to the subject matter and learning task



The types of content are very similar to the knowledge dimensions of the revised cognitive Bloom's taxonomy [3]. The Use performance level focuses on an ability to use an existing framework to process input. The Find performance level focuses on the ability to create a new framework through the adaptation of existing rules. This has similarities to the Apply and Create categories of the revised cognitive taxonomy.

2.5 Discussion of existing taxonomies

By far the most widely used of the taxonomies reviewed above is the original work by Bloom et al. Its strengths are that it is based on extensive analysis of test items, its simplicity, and its identification of distinct, recognizable aspects of the cognitive domain. Instructors have taken it to mean that they can assess comprehension, application, analysis, synthesis and evaluation and that this hierarchy maps onto a grading scheme. The weaknesses of the original Bloom's taxonomy is that the categories have not always proved easy to apply, that there is significant overlap between the categories and debate about the order in the hierarchy of analysis, synthesis and evaluation. In addition, its simplicity means that each category combines different types of cognitive activity.

There are many variants of the original Bloom. There is evidence that the revised category names used by Anderson et al have been adopted by instructors but it is not clear that the added complexity of distinguishing aspects of the cognitive domain such as procedural and metacognitive knowledge outweighs the simplicity of the original scheme. Facione's work is similar in its approach to improving on Bloom.

The work of Niemierko, Tollingerova and Bespalko has strong similarities to Bloom but produces two separate dimensions related to knowing and applying. This addresses the difficulty of regarding Bloom's categories as a single hierarchy but does not map so nicely onto a six or seven point scale. Component display theory identifies essentially the same dimensions but is specialized for use in computer-based instruction.

SOLO is very different to the other taxonomies reviewed above because it deals with the content of the learner's response to what is being assessed. Its holistic approach means that it can be used to assess performance in the affective and psychomotor, as well as cognitive, domains. By comparison with Bloom, it may be regarded as giving less guidance to instructors because is does not map onto categories of cognitive performance that can be singled out for assessment. Its strength is in encouraging a holistic approach that supports deep learning, its weakness that there is not yet much reported experience of using it for assessment in a range of subjects.

3. THE USE OF TAXONOMIES IN COMPUTER SCIENCE EDUCATION LITERATURE

3.1 Existing Literature on Taxonomies for Computer Science

A number of papers have explored how various generic taxonomies can be applied to computing topics. In particular, there are three ways in which such taxonomies have been applied: to the design of courses at various levels of granularity in time, the design of teaching, learning and assessment materials, and, finally, the analysis of student responses to exercises. In this section, we review work on these topics.

3.1.1 Design of Courses

Some authors propose using these taxonomies for the design or evaluation of courses. Indeed, the notation of educational objectives was the original purpose of Bloom's taxonomy. This can be at a number of different granularities: it could be used for describing student progress through a single topic, through a course, or through a whole degree programme.

Howard et al. [23] propose to clearly identify goals for every lesson, and to assign them to a given level of the taxonomy. Most lessons have a number of knowledge goals, but achieving other levels varies during the course. Plotting the highest level of each level in a graph shows the evolution of the course according to knowledge depth. Scott [65] states that assessment should measure the level achieved by each student, and the grade should depend on his/her achievement. In particular, he notices that his teaching has been covering levels 3 (application) and 6 (evaluation). Buck and Stucki [11] outline an inside/out pedagogical approach based on Bloom's taxonomy for cognitive development. This framework allows students to comprehend the basic concepts before they are asked to apply them.

Doran and Langan [17] report on a project that implemented a cognitive-based approach (using Bloom's taxonomy) to the first two years of a computing degree, using strategic sequencing (spiral) and associated mastery levels of key topics. The project also investigated the use of structured closed labs, with frequent feedback and early use of teams. They used course micro-objectives mapped to specific levels in Bloom's taxonomy. Machanick [43] describe his experience of applying Bloom's Taxonomy in design three different courses.

Some applications have applied the taxonomy across a programme of study for a degree. For example, Sanders & Mueller [64] discuss the redesign of the curriculum at his university to bring material that is concerned mainly with lower Bloom levels to the early years of a degree programme, and vice versa. In other areas, Bloom's taxonomy has also been used to redesign whole curricula. In particular, Reynolds & Fox [62] extend a curriculum in Information Technology based on the ACM Curriculum'91 to include new knowledge units and describe they fit it in Bloom taxonomy levels. In the same area, Azuma et al. [4] extend this taxonomy in order to apply it to Software Engineering. Manaris & McCauley [44] presented one possible implementation of the HCI curricular guidelines

included in CC'01. This implementation employs Bloom's taxonomy to identify levels of student competence for each of the learning objectives.

Oliver et al. [51] discuss the idea of a *Bloom Rating* for courses of study. The course assessments are analysed by instructors and the level in Bloom's taxonomy that the assessment is designed to engage the students at. These are then averaged for all of the assessments on the course, and this is termed the Bloom rating. This is then applied to looking at how courses develop in the cognitive demands that they make on the students over the three years of their degree programme. They note that some modules early in the degree programme have a high rating, and some towards the end have a low rating.

This paper makes a number of assumptions about the use of Bloom's taxonomy. Firstly, that the course should develop students' cognitive skills over the (three) years of the course, engaging students at a low cognitive level at the beginning of the degree and working towards the higher levels towards the end of the degree. There is also the assumption that an assessment works at one particular level. A danger with this is that becomes normative, and that it is used as a "quality measure" – the higher the Bloom rating, the better the course.

Johnson and Fuller [27] report on two studies of computer science courses carried out by students in the first year of computer science studies within a university: a panel of assessments rated by instructors, and interviews with the instructors on each course. A significant conclusion from these studies is that the most significant level for many of the courses studied is the *application* level; applying techniques to the creation of artefacts would seem to be at the core of what the study of computing is about. However, for complex application problems students need to use skills that would be classified at the analysis/synthesis/evaluation levels. The authors propose a new level of "higher application" for subjects such as computing. This encompasses cognitive activity that is aimed a solving a problem, yet which needs the traditionally "higher level" skills that engage students at the analysis/synthesis/evaluation level.

A recent paper by Kramer [31] identifies *abstraction* as a core skill that is important for many areas of computer science. The author discusses Piaget's model of cognitive development, which consists of four stages: sensorimotor, pre-operational, concrete operational, and formal operational [54]. His argument is based on studies that show that a significant percentage of the general population do not develop this final stage in the taxonomy: they do not progress to the stage of making significant use of the formal operational processes. Following on from this, he argues that getting students to this stage is a prerequisite for the students studying many aspects of computing, and that we should devise courses that ensure that students reach this stage of general cognitive engagement with material that they encounter *before* teaching most computing topics, or that we could use measures of abstraction ability as a way of selecting students for computing courses.

Finally, Rademacher [56] reports research in progress includes the conceptual development of a model and metrics to determine and classify the level of cognition and added value included in selected knowledge management (KM) systems. He joins Bloom's Taxonomy of Cognitive Objectives and Greenwood's Six C's of the Knowledge Supply Chain in order to contribute a new approach for assessing the role of knowledge management systems including value, skill sets, learning, modeling, and media.

3.1.2 Design of Teaching Materials and Assessments

Another way in which these taxonomies are used is in designing teaching materials and assessments. For example, structuring materials to help students to move through a taxonomy, or structuring assessments so that they assess a wide range of levels of engagement with this material.

A number of authors have discussed how learning taxonomies can be used for assessment design. Lister [38] notes that typical assessments in introductory programming leap straight into higher levels of Bloom's taxonomy, and presents a course design and examples of assessments that move students through the Bloom hierarchy. Thompson [70] reports on the use of the SOLO taxonomy to structure the marking scheme for a programming course, and in particular using this taxonomy to help students understand the grade that they have been assigned. Farthing et al. [19] discuss the design of a new kind of multiple-choice question (*permutational* MCQs) that can be used more readily than traditional questions to assess higher-level skills.

Lahtinen and Ahoniemi [36] are concerned with the use of taxonomies for the design of visualizations to help students understand programming not only in the elementary cognitive levels but to support their progress further also. They look at each level of Bloom's taxonomy, and discuss the kinds of visual material that would be relevant to presenting and interacting with material at each level resulting into a categorization of program visualization examples. Naps et al. [49] make a comprehensive study about the educational effectiveness of visualizations for computer programming education. They identify a set of good practices that have proved to be educationally effective. Bloom's taxonomy is proposed as a standard framework that educators can use to measure such effectiveness. Ihantola et al. [25] have developed a taxonomy of algorithm visualizations: whilst not a "learning taxonomy" as such (it does not give a structure for how students' development is meant to be guided by these visualizations) it could be used alongside such a learning taxonomy to investigate the match between students' development as learners and the technology required to support that development.

Some authors have designed software tools to assist at some level. Thus, Kumar [34] has developed a set of applets (named "problets") to assist at the application level for well-delimited topics. Each problet allows randomly generating instances of a problem involving a concept, a question to be answered and some kind of visualization or interaction to help solving the problem.

Buck and Stucki [11] extend the JKarelRobot environment to give support to all the levels in Bloom's taxonomy. For instance, students are continuously asked the next statement to be executed by Karel. At the end of the run, they are given a score that shows their competence at the comprehension level. Ala-Mutka [2] reports a different automated assessment approach. Facts: there exist different objectives and evaluations but these

objectives are not reached. The reasons are: There aren't obvious and joint criteria and the design of tasks is not careful. A possible solution is to design the objectives, the tasks and the assessments with some obvious criteria based in Bloom's Taxonomy.

Hernán-Losada et al. [21] describe insecurities and ambiguities that they found in applying taxonomies to the design of educational tools. They may classify difficulties into two classes: terminology and the inherent complexity of programming itself. They propose a guide to use the taxonomy within the Computer Science. Moving on from this, in their more recent paper [22] they describe their experiences with designing and developing learning tools inspired by the taxonomy of Bloom. They present a generic framework for the design of these applications and describe the tools developed for the learning of object-oriented programming.

3.1.3 Analysis of Student Responses to Exercises, and Measuring Student Progress

Whalley et al. [72] investigate the results of applying the Bloom and SOLO taxonomies to analysing the results of a programming exercise that was carried out by students at a number of universities. Nine of the questions in this exercise were multiple choice, the final was a free-text question that required students to give an English description of a piece of code. The conclusions of this paper are that the difficulty of these questions correlates strongly with their placement on the taxonomies (in that most students can tackle the lower-rated questions, a subset of those can perform on the higher level questions, then a subset of them on the highest). A particular item of interest is the free-text question that was asked at the end. The authors use SOLO to analyse the responses to these questions. This is carried further in [43] where they analyse the responses to this question and to a further question, related to classifying programs and investigating similarity between programs, and examine students responses using the SOLO taxonomy.

Lister et al. [42] report on the authors use of the SOLO taxonomy to describe differences in the way students and educators solve small code reading exercises. Data was collected in the form of written and think-aloud responses from students (novices) and educators (experts), using exam questions. During analysis, the responses were mapped to the different levels of the SOLO taxonomy. From think-aloud responses, the authors found that educators tended to manifest a SOLO relational response on small reading problems, whereas students tended to manifest a multistructural response. These results are consistent with the literature on the psychology of programming, but the work in this paper extends on these findings by analyzing the design of exam questions.

Lister and Leaney [39,40] also notice that typical programming assignments correspond to level 5 (synthesis). Instead, they group the six levels of the taxonomy into three pairs, so that achieving a level in a given pair yields the corresponding A, B or C grade. In addition, they identify grading practices adequate to each pair, namely lab exercises and exams, multiple choice exams, assignments, projects, and peer review. These ideas have been applied by Box [9], in particular emphasizing the way in which taxonomies can be used to provide a transparent means by which assignments can be explained to students and students can understand their grade and how performance fits into overall progress on courses. In particular, this paper gives comprehensive guidance to lecturers who are considering using Bloom-style structuring for their assessments. Cukierman and McGee Thompson [16] report on the use of Bloom's taxonomy directly with students, in order to help students devise learning strategies to help with their learning of topics in computer science.

The paper by Burgess [13] reports on the author's experience with using Bloom's taxonomy in marking assessments. The grade given to an assessment depends on the level in Bloom's taxonomy that the student's response suggests that that student is working at.

Buckley and Exton [12] review Bloom's taxonomy as a richer descriptive framework for programmers' knowledge of code and illustrates how various software maintenance tasks map to knowledge levels in this hierarchy. A pilot study (with 2 students) is presented showing how participants' knowledge of software may differ at various levels of this hierarchy.

4. EXAMPLES OF THE USE TAXONOMIES IN SOME CANONICAL COMPUTER SCIENCE COURSES

The interaction between typical computing learning outcomes and taxonomies can be further illustrated through examples. This subsection presents three such examples, chosen to be typical of courses that appear in a wide range of computing curricula. One is a first year course, the second is from material that is often given at an intermediate level and the third demonstrates features of final year courses. They are all based on actual courses but have been adapted to suit the needs of this paper. The discussion covers the use of Bloom's taxonomy of the cognitive domain and the SOLO taxonomy, because these are the only ones that we found being used in practice in the computer science education literature. In addition, there is some consideration of Bloom's taxonomy of the affective domain because this could improve constructive alignment between the values instructors want to instill and the ways we assess computing students.

4.1 Introductory Programming Example

4.1.1 Description of course

This is typical introductory object-oriented programming course. It lasts for a single semester and takes an objects-first approach to teaching Java programming, closely following a well-known textbook. The students have lectures and classes (labs) each week. The lectures, which are optional, introduce new concepts. Students are expected to do programming exercises in the class sessions and finish these off in their own time. Some of the class exercises are marked and these marks contribute 20% of the final course result. The main assessment for the course is currently a closed book examination that contains a mixture of multiple choice questions and essay answers.

4.1.2 Learning Outcomes

At the end of the course students will be able to

- Use an object-oriented programming language to write programs.

- Discuss the quality of solutions through consideration of issues such as encapsulation, cohesion and coupling.
- Recognise and be guided by social, professional and ethical issues and guidelines

4.1.3 Assessment using Bloom in the cognitive domain

Bloom's taxonomy in the cognitive domain is conventionally used to assess the first two learning outcomes given above. A typical approach is to write assessment items that are intended to assess at a single level and then to award some fraction of the total number of marks available, depending on how complete the student's response is seen as being. It is relatively unusual to have tasks that are seen as giving students the chance to respond at more than one level, along with assessment criteria indicating which level the student is seen as operating at.

4.1.3.1 Example 1

Consider the following class definition.

```
public class Car
{
	public int numberOfSeats;
	private String model;
	private int engineCode;
	public Car(String model)
	{
		model = model;
	}
	public int getSeats()
	{
		return numberOfSeats;
	}

	private String getModel()
	{
		return model;
	}

	public void setEngineCode(int code)
	{
		int n = code * 2;
		if(code >= 100) {
			engineCode = n;
		}
		else {
			engineCode = code;
		}
	}
	}
```

Decide which statement is correct (A, B or C). Only one statement is correct.

Accessors / mutators

(a) The method `getSeats` is an accessor method.

(b) The method `getSeats` is a mutator method.

(c) The method `getSeats` is both an accessor and a mutator method.

Discussion This test item could be assessing recall if it is using an example that the students have seen before. If they have not, it could be a simple example of application of a rule. A conscientious, or over anxious, student could have come across this example before even if it was not used in lectures. A student who did not bother to go to lectures may be applying thise rule from first principles, even if the examiner expects students to be using recall. It is thus hard in practice to determine which of these two Bloom cognitive levels a student is performing at.

4.1.3.2 Example 2

In designing an application, the concept of coupling is important. One guideline states that you should have weak coupling. What is coupling, and why should you have weak coupling?

Discussion This tests whether students have reached the "explain" level. In the unlikely event that the reasons for weak coupling have not been spelt out in lectures, it could be at a considerably higher level.

4.1.3.3 Example 3

Write a method to calculate the winnings of a lottery ticket with three integers, **`a`**, **`b`** and **`c`** on it. The header of the method is

```
public int lotteryTicket(int a, int b, int c)
```

If the numbers are all different from each other, the method returns 0. If all of the numbers are the same, the method returns 20. If two of the numbers are the same, the method returns 10. For example:

lotteryTicket(1, 2, 3) → 0

lotteryTicket(2, 2, 2) → 20

lotteryTicket(1, 1, 2) → 10

Write a full implementation of this method.

Discussion The instructor is likely to expect this to be straightforward example of apply.

4.1.4 Assessment using SOLO

Example 1 above focuses on a single piece of information, ie recognizing the naming of an accessor method. This means that it can be used to assess at the unistructural level.

4.1.4.1 Example 4

Provide two examples of loop constructs that can be used in a method to calculate the minimum value in an array. The header of the method is

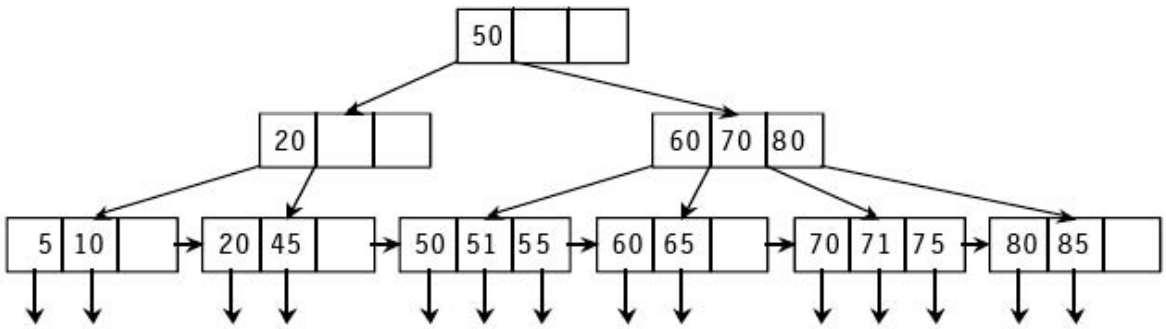
```
public int min(int []a)
```

Discussion this test item requires identification of two distinct loop constructs but not necessarily working code. This means that it can be used to assess at the multistructural level. If the question asked the student to write a routine that calculates the minimum value then it is targeting a relational response since to develop working code requires an understanding of how different constructs work together.

4.1.4.2 Example5

In plain English, explain what the following

For each of the following modifications, show the result B^+-tree obtained by applying the modification to the B^+-tree shown below. Suppose that the maximum fan-out is 4. (Always start with the B^+-tree shown below; do not apply the modifications to the result of previous modifications.)



(a) Insert 21.
(b) Delete 50.
(c) Insert 79.
(d) Delete 10.

Figure 1. Example 8.

segment of code does:

```
bool bValid = true;
for (int i = 0; i < iMAX-1; i++)
{
   if (iNumbers[i] > iNumbers[i+1])
   {
      bValid = false;
   }
}
```

Discussion This seeks a relational response in the sense that the student needs to recognise what is being performed as a whole (relational response) rather than describing the actions of the individual statements (multistructural response); see [40,70,72].

4.1.4.3 Example 6

Performance at the extended abstract level requires students to generalise their knowledge. An example of testing for this could be as follows: the students have been taught how to use an ArrayList. They are now asked to implement code using the Java library LinkedList class. This expects them to generalise the knowledge of working with one collection type and apply it in a near context.

4.1.5 Assessment in the Affective Domain

The learning outcome "Recognise and be guided by social, professional and ethical issues and guidelines" represents an area of learning in which instructors want students to take what they have learnt to heart, not simply to be able to play back what has been told to them. To provide constructive alignment between learning outcome and assessment, it is necessary to assess in the affective domain. The problem is that there is no time for this learning to be embedded, so it is not very feasible to assess it during this module. The answer may be to move the assessment of the affective dimension to a later course.

4.2 Databases Example

4.2.1 Description of course

This course is an introduction to the principles, use, and applications of database systems. It assumes no previous knowledge of databases or database technologies. Topics include: an introduction to relational database systems, relational database model, entity-relationship model, relational algebra, SQL, relational design, and advanced topics such as relational query evaluation, XML databases, and fundamentals of transactions and concurrency.

4.2.2 Learning Outcomes

This course contributes to the development of the following capabilities:

- Enabling Knowledge: Fundamental database concepts including analyzing, designing, defining, constructing and manipulating relational database systems.
- Problem Solving: Ability to design and implement database solutions for various application areas and to build queries for users' needs, based on analysis of data modeling problem specifications.
- Critical Analysis: Ability to analyze data modeling problem specifications and derive alternative conceptual models that represent the problem in different perspectives leading to alternative database designs.

4.2.3 Assessment using Bloom's taxonomy in the cognitive domain

4.2.3.1 Example 7

The INSERT statement provides an optional clause to list the columns that you are inserting values into. Why is it prudent to list the columns when you are developing code for a production system?

Discussion This invites students to describe the syntax of an INSERT statement and infer what can go wrong. This is the Comprehension level of Bloom's cognitive taxonomy. However, students who do not know the syntax but have learnt by trial and error that not listing the columns can produce unexpected results may answer at the lower level of Remember.

4.2.3.2 Example 8

See Figure 1.

Discussion Most database courses drill students on this kind of problem, so the question requires Application of known rules. Note that no explanation is required. Many instructors would consider that this would make the question more difficult, even though Comprehension, and Explaining in Anderson et al's revision of Bloom's taxonomy, comes at a lower level than Application.

4.2.3.3 Example 9

A database contains the following tables:

MOVIE(movieID, title, yearReleased, genre, ratingCode, nationality)

RATING(ratingCode, ratingDescription)

PERSON(name, DoB)

MOVIE_PERSON(movieID, name, role)

where role can take the values "Director", "Producer", etc.

Write a query to return the title, rating, and year released of all movies released from 1970 – 1995 inclusive that were directed by Quentin Tarantino, Ron Howard, or Brian DePalma. Movies should be listed from most to least recent with titles listed alphabetically for each year.

Discussion This type of question typically presents a new scenario to the students, so they are expected to operate at the Analysis level to solve it.

4.2.3.4 Example 10

Roger Ebert, a well-known movie critic, wants to compare directors across ratings and genres to see if there are any trends (e.g., do certain directors typically choose movies from a particular genre with particular ratings?). Using the tables in example 10 above, write a query to help Roger analyze the directors who have released one or more movies since 1960. Specifically, list each director along with the genre, rating description, and the number of movies the director has directed in the given genre with the given rating. However, keep the amount of data manageable by only including rows with more than 10 movies. List your results from highest to lowest number of movies. If multiple rows have the same number of movies then list the director, genre, and rating description alphabetically.

Discussion This is a more complex example of analysis. It falls short of Synthesis, or Creating in the revised Bloom's taxonomy, because the problem is very self contained and there is effectively a single right answer. If the student had to find out about the world of movies as well as about databases, it would required synthesis.

4.2.3.5 Example 11

For each schedule below, tell whether it is conflict-serializable. If yes, also tell:

- Whether it is recoverable;
- Whether it avoids cascading rollbacks;
- Whether it is possible under *strict* 2PL.

(a) $T1$.write(B), $T2$.read(A), $T2$.write(A), $T1$.read(A), $T1$.write(A), $T1$.commit, $T2$.commit

(b) $T1$.write(B), $T2$.read(A), $T2$.write(A), $T1$.read(A), $T1$.write(A), $T2$.commit, $T1$.commit

(c) $T1$.write(B), $T2$.read(A), $T2$.write(A), $T2$.commit, $T1$.read(A), $T1$.write(A), $T1$.commit

(d) $T1$.write(B), $T2$.read(A), $T1$.read(A), $T2$.write(A), $T1$.write(A), $T2$.commit, $T1$.commit

(e) $T2$.write(B), $T2$.read(A), $T2$.write(A), $T1$.write(B), $T2$.commit, $T1$.read(A), $T1$.commit

Discussion This also requires analysis but falls short of evaluation.

4.2.4 Assessment using SOLO

Example 7 above seeks a unistructural response because it deals with a single construct. Example 8 is multistructural because knowledge of both Insert and Delete constructs is required but they are used independently. Examples 9 and 10 target a relational response because the student has to understand how SQL syntax can be applied to her or his analysis of the problem. Example 11 is also seeking a relational response.

4.3 Computing Professionalism Examples

Professionalism within computing is a topic of concern to many professional organizations (IEEE/ACM, BCS etc). These organizations have sought to make professionalism an explicit learning objective (instructional modules) at the university-level. Within computing, this often involves some form of work-based learning. The question of concern is how to assess professionalism? Instructors have often relied on written reports to assess the student's ability to *apply* professional concepts. Additionally, many instructors have attempted to assess professionalism through the use of peer, employee and self evaluations.

4.3.1 Description of course

A course in computing professionalism covering topics concerning the social impact, implications and effects of computers on society, and the responsibilities of computer professionals in directing the emerging technology. Relevant professional skills are explored via active-learning activities such as business writing, oral presentations, debates, job hunting and interviewing, professional etiquette, critical thinking, and peer reviewing. An extension to this course gives students the opportunity to apply their skills in consulting capacity, working with real clients to solve their problems.

4.3.2 Learning Outcomes

Students completing this course should be able to...

- review and analyze the effects—both anticipated and observed—of the insertion of computer technology into many aspects of society;
- combine their understanding of technology's effects with their personal values, to express and carry out ethical behavior with respect to computing and its impacts, including an ability to articulate and weigh the pros and cons associated with diverse ethical positions;
- identify, analyze, and act upon work situations that have potential ethical, legal, or other professional implications;
- produce written documents of varying type and size in a competent and professional fashion, including the ability to review and critique colleagues' work;
- design and deliver an interesting, concise, and relevant oral presentation with technical content.

Students completing the extended course will

- Be able to apply the concepts and techniques required to build software systems to meet the needs of small enterprises
- Have developed their own computing professional identity through applying the ACM/IEEE code of ethics
- Interact "professionally" with a client through meetings, written reports and email.

4.3.3 Assessment using Bloom's taxonomy in the cognitive domain

4.3.3.1 Example 12 ***A review of a technical article***

Following reviewing and editing guidelines, students are asked to analyze and critique an assigned article, including providing an answer to questions dealing with the organization and writing style of the article.

Discussion This requires students to Evaluate in the cognitive domain. There is also an element of Synthesis (Creating in the revised taxonomy), particularly if the students are expected to extend the review to their own discussion of the topic of the article.

4.3.3.2 Example 13 ***Group Debates***

The debates are intended to sharpen the student's skills to adopt and support one or more viewpoints on an issue about ethics or professionalism in the workplace. The class is broken down into groups of 4-5 students. Each team will choose an ethics topic and write a scenario that raises issues associated with this topic. Teams are instructed to choose topics that have believable arguments both pro and con.

General topics to consider include Special needs, ADA requirements, Universal accessibility, Consideration of public risks in system development, Internet censorship, Competitive intelligence or industrial espionage, Intellectual rights, copyrights, & patents, Privacy, National missile defense system, Protection of the environment or ecology, Ethics of medicine or biotechnology, Scientific fraud or plagiarism, Hackers, Professional and legal liability for defective information or software, Viruses, worms, and other "malware", Technological obsolescence (losing jobs to automation), Cryptography and public encryption, Whistle-blowing.

Discussion This allows students to demonstrate skills of analysis, synthesis and evaluation. Note that because they are asked to take a stance for the sake of debate, they cannot be assessed in the affective domain.

4.3.3.3 Example 15

Proposal to a hypothetical work group about a professional issue: This assignment takes place in four phases. The first deliverable is a two-page (500 words) plan for how the student is approaching the proposal-writing process. The second will be a first draft of an 8-page proposal (approximately 2000 words) researched and written according to the earlier plan. The third deliverable is review of another student's proposal. The fourth deliverable is a final draft of the proposal, in which the student makes revisions and responds explicitly to the review feedback.

Discussion This gives students excellent opportunities to demonstrate synthesis and evaluation.

4.3.4 Assessment using the SOLO taxonomy

If the SOLO taxonomy is used in assessing professionalism then for a unistructural assessment, a single professionalism attribute would be assessed. A multiustructural assessment would seek to assess to professionalism attributes in a way that was independent of each other. A relational assessment would focus on how the professionalism attributes are integrated together in the assessment exercise. An extended abstract assessment would seek to observe professional attributes that are being interpreted in new ways.

In utilizing the SOLO taxonomy, it is not simply the professionalism attributes that can be assessed. In assessing at the relational or extended abstract level, it is possible to assess how professionalism interacts with or relates to other more technical attributes.

4.3.5 Assessment in the Affective Domain

The learning outcomes of the extended course described above are concerned with the development of professional attitudes and values as well as with cognitive skills. These can be measured using a variety of instruments. One is a reflective log, in which students are asked to report their feelings and motives and to evaluate their own performance in the consultancy role. Another instrument is the instructor's observation: was the student proactive in working professionally or was nagging required to ensure that tasks were completed punctually and to a high standard? Finally, feedback from the clients has an important role in determining whether the student's professional values and commitment are demonstrated under all circumstances.

5. WHAT IS SPECIFIC ABOUT COMPUTER SCIENCE

The learning taxonomies discussed in sections 2 and 3 are generic, implying that the types of learning and the ordering of the hierarchy are constant across subjects. However, this may not be the case. For example, in applied subjects such as computing, a principal learning objective is the ability to

develop artifacts (in computing, pieces of software) [30]; by contrast, instructors in other subjects (such as English Literature) place more emphasis on skills of critique and less on producing artifacts (such as novels). It could therefore be argued that in applied subjects, Application encompasses Synthesis and Evaluation, rather than being a lower level skill. It is notable that the recent ACM overview of computing curricula [28] refers to *performance competencies* rather than *learning outcomes*, reinforcing the perceived importance of Application.

We can also distinguish between disciplines in which there is an emphasis on learning through *interpreting* and those in which learning is predominantly achieved through *doing*. Economics and Theology could be seen as examples of the former, Dance and Music performance of the latter. This is not to suggest that Economics and Theology do not require their students to do in the sense of repeatedly writing essays; however they are learning about the practice of the subject rather than running an economy or developing a new religion. Computing students are expected to do a lot of learning through doing, whether it is learning about software engineering by developing systems of increasing complexity, learning about networking by implementing protocols or learning about group dynamics by working in teams.

There are several other characteristics that apply specifically to computer science as discipline. First, and perhaps foremost, studying processes and problem solutions is very central to, if not the essence of, computer science. One could say that solving problems and producing an effective and efficient solution is the core goal of a computer science professional. Computer science centrally involves modeling the real world, representing domains of the most varied nature and complexity, representing knowledge in general and dealing with processes and solutions for problems in such domains.

In order to address the complexities of the problems and domains, there is an essential need to abstract and decompose problems into subproblems and modules. Abstraction, modularity and reusage of previous solutions constitute essential abilities needed by any computer science researcher or professional.

Other characteristics of computer science are creativity and openness to novelty, considering that they are inherently related to finding solutions to problems. It is also worthy of notice that computer science is becoming more and more multidisciplinary, and hence professionals and academics need good communication skills not only among themselves but also with experts in other disciplines.

The following list of keywords encompasses what this working group considers to be intrinsic characteristics of computer science. Clearly, a comprehensive learning taxonomy should be useable for assessment of all of them.

Intrinsic characteristics of computer science:

Problem solving

Domain modelling

Knowledge representation

Efficiency in problem solving

Abstraction/modularity

Novelty/creativity

Categorization

Communication skills with experts in other domains

Adoption of good practice in software engineering

This final feature of computing reflects the need to develop professional skills and values. It is not enough that students should know what constitutes good programming style; we want them to have taken this to heart so that they instinctively write elegant code whenever they work on a piece of software, not just when marks are explicitly available for doing so. Similarly, any intended learning outcome relating to the ACM/IEEE or other

professional code of conduct ought to go beyond "Knows about the code of conduct". We want students to respond positively to it by internalizing it and making it part of their personal set of moral and ethical principles, so that they automatically behave according to its precepts, even under challenging circumstances

6. A NEW TAXONOMY FOR COMPUTER SCIENCE

In this section we present a new taxonomy designed to be suitable for computer science and engineering, especially for learning programming (in the broadest meaning of the word). We also present a novel way to apply any existing taxonomy which better deals with modularity and increasing levels of abstraction, aspects that typify engineering and computer science in particular.

6.1 Two Dimensional Adaptation of Bloom's Taxonomy – The Matrix Taxonomy

The intent of the proposed taxonomy is to provide a more practicable framework for assessing learner capabilities in computer science and engineering. The immediate target for this work is computer programming, but we feel the taxonomy is applicable to other fields of engineering in which practitioners produce complex systems. It is meant as a partial solution since (among other things) it does not address the affective domain, only indirectly deals with abstraction skills, and incompletely handles structural relationships in the content.

The inspiration for this taxonomy was research [41,73] indicating that comprehension of program code and the ability to produce program code are two semi-independent capabilities. Students who can read programs may not necessarily be able to write programs of their own. And the ability to write program code does not imply the ability to debug it. Robins et al. [63] describe this independent interpretive skill as the ability to distinguish the intended behavior of the program from the actual behavior of the program.

Although a review of the literature reveals a wide range of possible candidates, only Bloom's taxonomy of the cognitive domain appears to be widely used in computer science course and assessment design. Its main strengths are that the levels are reasonably easy to understand and there is a developing literature, reviewed above, on how to use it to devise test items. Thus we felt it would form the most natural basis for our proposed taxonomy.

We used the revised version of Bloom's taxonomy [6] which responded to problems with the linear approach at the higher levels. It provides a level of creation (Higher Application) which requires competency at all the previous levels and one that does not (Create). In order to visualize this distinction and the semi-independent skills of reading and writing program code, our taxonomy employs a two dimensional matrix with an adaptation of Bloom's taxonomy which is presented in Figure 2.

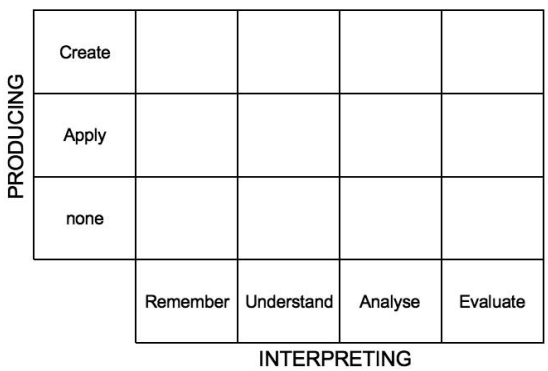


Figure 2. A graphical presentation of the two dimensional adaptation of Bloom's taxonomy.

The dimensions of the matrix represent the two separate ranges of competencies: the ability to understand and interpret an existing product (i.e. program code), and the ability to design and build a new product. Levels related to interpretation are placed on the horizontal axis and levels related to generation are placed on the vertical axis, with the lowest levels at the lower left corner. The names of the levels are from the revised version of Bloom's, as we feel they are sufficiently unambiguous. It is understood that students traverse each axis in strict sequence. For example, it is not possible to begin to do synthesis (Create) until there is some degree of competency through the Apply Level.

6.1.1 Applying the taxonomy – traversing the matrix

The matrix should be especially useful for instructors needing a marking grid for their students. Also it rather clearly illustrates all the different learning paths students may take, as discovered in recent work by Lahtinen [37].

Different students take different "learning paths" in the matrix taxonomy. For instance, when a student learns a new programming concept he first achieves the knowledge of this concept. At that point the student is in the cell (the state of) "none/Remember" shown in Figure 2. If this student continues with learning by imitating a ready example of a program but without deep understanding of the concept, they will achieve the state "Apply/Remember", i.e. applying/trying to apply the concept without real understanding, with trial and error. This behaviour is illustrated in Figure 3. If instead of imitating, the student decides to first find more information on this concept, as from a book, they might proceed to the cell "none/Understand" to the right of the initial cell. This means that the student is not yet able to produce program code, but he might already understand the meaning behind this concept.

A competent practitioner of a concept would be placed in the cell "Create/Evaluate", which means that he is able to perform at all the competency levels in the matrix. This can also be identified as the level Higher Application [27] and can be reached through different paths as shown in Figure 6.

However, there are students who attain only some of the competencies. For instance, *the theoretical students* identified in a cluster analysis study [37] may be placed in the cell "none/Evaluate" which means that they are able to read program code, analyze, and even evaluate it, but cannot yet design a solution or produce program code. This is not the most common pathway for students to follow, but these students have only proceeded in the horizontal direction as shown in Figure 4.

The same study revealed another group, *called the practical students*, who could be placed in the cell "Create/Understand" of the matrix. Being in that cell would indicate the ability to apply and synthesize without the ability to analyse or evaluate even their own program code. This behaviour is illustrated in Figure 5. The problem for these practical students is in not being able to debug their own solutions when they encounter errors.

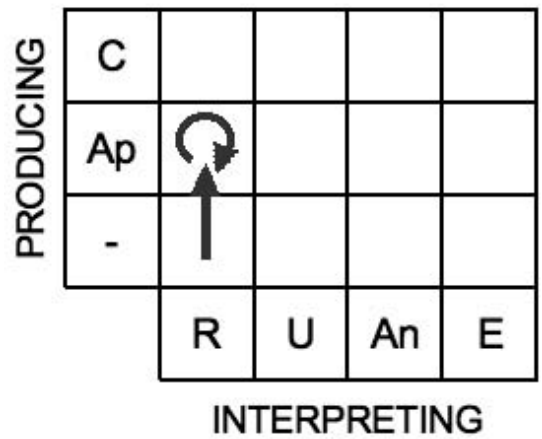


Figure 3. A student trapped in trial and error approach

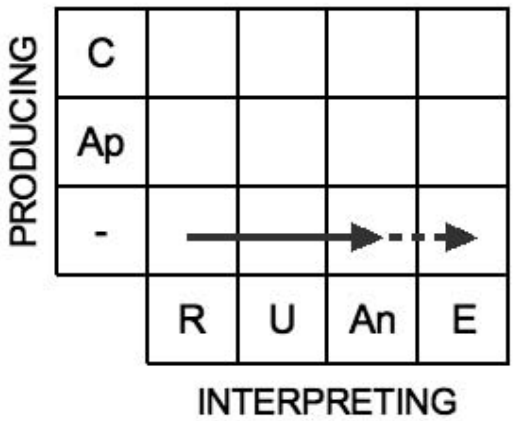


Figure 4. The pathway of the students who attain only theoretical competencies.

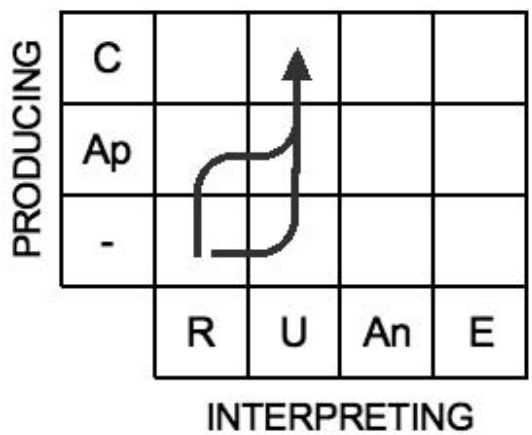


Figure 5. The pathway of the students who attain only practical competencies.

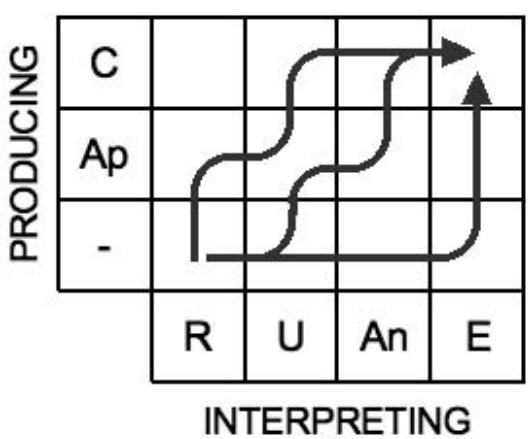


Figure 6. The goal, "Create/Evaluate" or Higher Application, can be reached through different pathways.

Mapping Programming Activities to the Matrix

We provide a mapping from a set of computer programming activities to the cells of the matrix in order to illustrate the discriminatory power of the proposed taxonomy for this subject area. This is done with a list of problem-solving activities related to programming collected as a reaction to difficulties encountered in using Bloom's Taxonomy. The activities shown in Table 1 are mapped to the cells of the taxonomy. See Figure 7.

Table 1 – A list of problem-solving activities related to programming

Solution Activity	**Description**
Adapt	modify a solution for other domains/ranges
Analyse	probe the [time] complexity of a solution
Apply	use a solution as a component in a larger problem
Debug	both detect and correct flaws in a design
Design	devise a solution structure
Implement	put into lowest level, as in coding a solution, given a completed design
Model	illustrate or create an abstraction of a solution
Present	explain a solution to others
Recognize	base knowledge, vocabulary of the domain
Refactor	redesign a solution (as for optimization)
Relate	understand a solution in context of others
Trace	desk-check a solution

To "adapt" a solution probably requires competency close to Create on the vertical scale and at least Understand on the horizontal scale, because modifying involves production and knowing what and how to modify requires understanding. "Apply" in the meaning of Table 1 may be as high as Create on the vertical axis since it calls for some creative ability, probably more than implied by the Apply level, in spite of its name. The position in the horizontal axis depends on the situation. To "debug" calls for a collaboration of both interpretation and building so should be high on both axes, perhaps in the cell "Create/Analyse". The ability to "design" naturally implies Create on the vertical scale and likely some degree of interpretation on the horizontal scale, though how much is uncertain.

"Refactor" and "Relate" are shown at the highest level of interpretation because both call for a deep understanding of the context of the problem and solution. We view "refactoring" as involving an improvement on the original design, thus admitting a possible placement even higher than "design".

To avoid belaboring the mapping example, we simply state that similar reasoning inspired the placement of the remaining activities. The point is that a mapping is feasible and does result in a fairly complete covering of the grid. Furthermore, most of these activities are general enough to be immediately applicable to other fields of engineering.

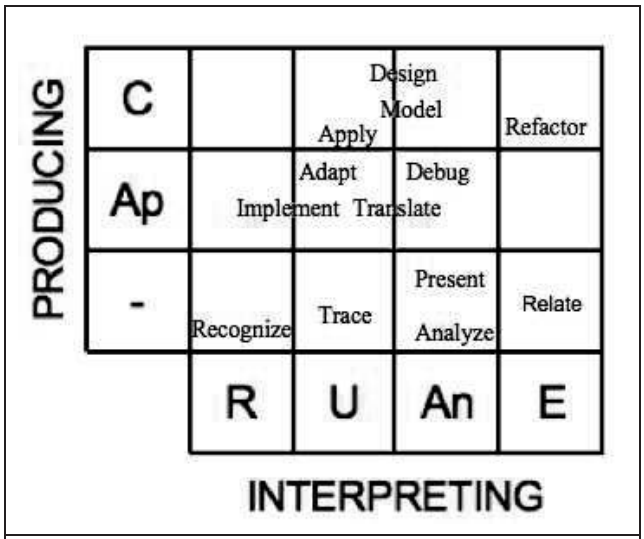


Figure 7. Mapping programming activities to the Matrix

Several of the solution activities may be amenable to assessment using the SOLO taxonomy, which considers the organizational complexity of the problem. This dimension is not at present well illustrated by our matrix, though it may be expected that SOLO levels generally increase as one goes from the origin to the upper right. Consider the activity "present": One would prefer the ability of presentation at the relational level of SOLO as opposed to uni- or multi-structural. "Design", "relate", and "model" are other activities we have identified for which SOLO is useful. In contrast, "implement" as defined in the table, involves applying a process to an otherwise completed design, and thus may be less related to skills involving complexity.

Many of the activities are related to the ability to work with abstraction, an ability that is vital for computer programming and has been discussed as an overriding argument for an alternative learning taxonomy [33]**.** Design, model, refactor, debug, and present may easily be seen to involve extensive consideration of abstractions. As examples, these activities may include as sub-activities the following: traversing levels of abstraction, mapping between levels (precision being essential for programming!), constructing new abstractions (with the attendant requirements of retaining needed detail and eliminating unneeded detail), adapting abstractions, and using abstractions as models of the original problem and/or solution.

A subject of some discussion in this working group was how to apply the matrix taxonomy to the affective domain. We have designed this taxonomy only for the cognitive domain but non-cognitive skills (e.g. social and emotional skills and the adoption of professional standards) also play a major part in programming practice. Internalization of professional practices is indeed an essential component of learning for computer programmers. Possibilities considered included extending the matrix in one or both directions by another level, or devising a companion matrix. Our overall feeling was that there is so little experience in computer science of assessment of values and attitudes that this would be premature. Krathwohl, Bloom and Masia's taxonomy for the affective domain [32] appears to be usable for courses aiming to develop professional values and we would like to encourage its adoption so that an evidence base can be accumulated.

6.2 Applying Taxonomies Iteratively - a Spiral Architecture for Applying a Learning Taxonomy

Robins at al. describe *a schema* as "a structured chunk of related knowledge" [63]. The student's learning goes through learning new schemas, modifying and combining them in order to produce new, more abstract schemas. Thus, the learning of programming could be seen as an iterative process. In the very beginning, the student is taught really simplistic and basic pieces of information and places to apply them. Instead of learning some things here and there, programming is a skill that is learned by building new information on top of earlier information. So in a way the basic pieces of information students are first struggling with become the bits and pieces they use in subsequent learning of new material. Compared to other cyclic learning styles e.g. the experimental learning style described by Kolb's Learning Cycle [30], the idea here is to proceed to a new level after each cycle.

The idea of a cognitive learning taxonomy can also be used in an iterative, spiral way. When the student is learning the basic concepts and the simplest subjects, he is going through the taxonomy in respect of that subject only. After having created a schema on that subject, he is then guided into a more abstract subject. When looking only at this new subject, the student is starting again from the lowest level of taxonomy—but now using the earlier material as a prerequisite.

The spiral process could be applied to Bloom's taxonomy, in that when the student is learning a new subject, his prerequisites—the materials to use in building new knowledge—have become his new basic knowledge, although the student has perhaps reached the level Create or Evaluate on those earlier subjects. Create could be described as the ability to combine one subject with others in order to build new solutions. This may also be seen when new solutions or subjects are learnt by building upon and integrating previous knowledge. This is easily seen to be true when considering that topics that are difficult and require in-depth analyzing by students are mere basic knowledge for expert programmers. Applying Bloom's taxonomy iteratively is illustrated in Figure 8.

Here is an example of a learning spiral: In the beginning a programming student is taught how to use a loop structure. He will go through all the levels of Bloom's taxonomy while learning it. He *knows* that a loop can be used for iteration; he *understands* how the loop works; he is able to *apply* a loop when told etc., eventually learning it thoroughly. After reaching the highest levels, the loop structure has become a tool for the student to use in subsequent programming. As the student is trying to learn how to sort an array, the loop can be seen as his basis knowledge upon which he is building his new knowledge. Later as the student is trying to implement a top-application[1] to

[1] The application that displays and updates sorted information about the top CPU processes

his own operating system, he will use the sorting of an array as a part of his base knowledge.

Traditionally programming has been taught starting with low levels of abstraction, moving on bit-by-bit to higher abstractions. For example, consider learning expressions, loop structures, functions, classes, design patterns etc. There are still many situations where one returns for more in-depth learning. Using a high level programming language itself establishes a starting level of abstraction, and using the objects-first approach immediately raises that level. The spiral approach with learning taxonomies must not be seen as going directly from bottom to top, but by seeing each round as thoroughly learning some new piece of information which is then used as a basis for the next round in the topic. It is of benefit to know how to write functions using C++ when one is trying to do something similar but more challenging with a lower level language such as Assembly, because then one already has knowledge of procedures, functions, parameters and return values.

The spiral application of a taxonomy is not limited to any particular taxonomy such as Bloom's. One round of the spiral (the learning of a new schema) could be described by any taxonomy suitable for describing students' abilities in that subject. For instance, the Matrix taxonomy proposed in subsection 1 could be applied in a spiral way. One learning path from the elementary level "none/Remember" to the Higher Application level "Create/Evaluate" can be seen as one round of the spiral. When rising to a higher abstraction level, the student starts his " learning path" once again from the lower left corner.

When trying to move up a level of abstraction (as in to start a new round of the spiral) the student may not have reached the Higher Application level "Create/Evaluate". To use his skills as a basic knowledge for the next, more abstract round the student may well be in one of the nearby cells, such as "Create/Analyse". While already progressing in the next round (with a more abstract subject), the student may eventually reach the "Create/Evaluate" state of the earlier level through his experience in using it. Thus the two rounds would in a way be followed in parallel for a while. On the other hand, if the student has taken one of the less desirable learning paths illustrated in Figures 3 and 4 (theoretical or practical only) and attempts to progress to the next round, he could be building his knowledge on misconceptions and may later face problems.

7. CONCLUSIONS AND RECOMMENDATIONS

Despite the wide range of taxonomies presented in this paper the Bloom's taxonomy of the cognitive domain seems to dominate the field of computer science course and assessment design. Though having many benefits, its principal weakness is that the levels do not appear to be well ordered when used to assess practical subjects such as programming. Our recommended solution is to separate Bloom's six levels into two dimensions, Producing (incorporating apply and create) and Interpreting (incorporating remember, understand, analyze and evaluate). This removes the strict ordering while retaining many of the concepts of Bloom's taxonomy. This generates a matrix that can be used to identify a range of different learning trajectories and hence to guide students in how to improve their skills and understanding.

Discussions with colleagues also exposed a lack of alignment between learning outcomes and assessment practice in the area of professionalism. Instructors bemoan students' lack of commitment to good engineering principles but fail to assess this, sending mixed messages to learners. This can be addressed by assessment in the affective as well as the cognitive domain. There no evidence in the literature of this being done, so the most sensible course would be to use an existing taxonomy for this purpose.

We recommend the use of our matrix taxonomy for the design and assessment of programming and software engineering courses. We also recommend that instructors and course designers use Bloom's taxonomy of the affective domain to achieve constructive alignment between their desire to produce computer scientists with professional attitudes and values and the messages they send through assessment tasks. Further work is needed to evaluate both these methodologies in computer science education.

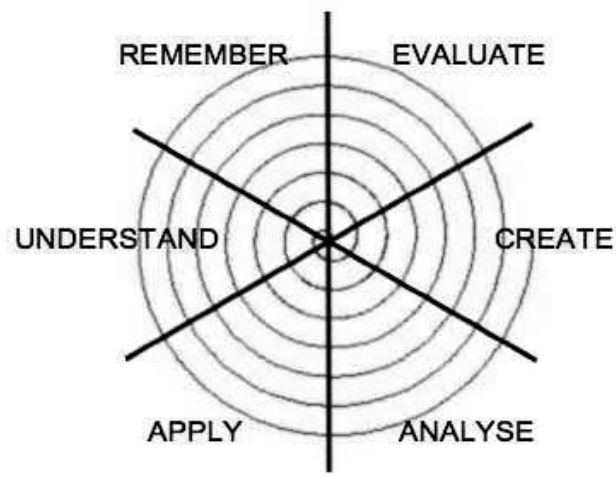


Figure 8. Bloom's Taxonomy as a Spiral Taxonomy.

8. REFERENCES

[1] Agencia Nacional de Evaluación de la Calidad y Acreditación. 2005. Título de Grado en Ingeniería Informática.

[2] Ala-Mutka, K.M. A survey of automated assessment approaches for programming assignments. *Computer Science Education* 15, 83-102, 2005.

[3] Anderson, L.W., Krathwohl, D.R., Airasian, P.W., Cruikshank, K.A., Mayer, R.E., Pintrich, P.R., Raths, J. and Wittrock, M.C., Eds. 2001. *A taxonomy for learning and teaching and assessing: A revision of Bloom's taxonomy of educational objectives.* Addison Wesley Longman, Inc.

[4] Azuma, M., Coallier, F. and Garbajosa, J. How to apply the Bloom taxonomy to software engineering. *Software Technology and Engineering Practice: Eleventh Annual International Workshop on,* 19-21 Sept. 2003, 117-122.

[5] Biggs, J.B. and Collis, K.F. 1982. *Evaluating the quality of learning: The SOLO taxonomy (Structure of the Observed Learning Outcome).* Academic Press, New York.

[6] Biggs, J.B. *Teaching for quality learning at university.* Open University Press, Buckingham, 1999.

[7] Bloom, B.S., Engelhart, M.D., Furst, E.J., Hill, W.H. and Krathwohl, D.R. 1956. *Taxonomy of Educational Objectives: Handbook 1 Cognitive Domain.* Longmans, Green and Co Ltd, London.

[8] Bologna Secretariat. Framework of qualifications for the European Higher Education Area, 2005.

[9] Box, I. Assessing the assessment: an empirical study of an information systems development subject. *Proceedings of the fifth Australasian conference on Computing education - Volume 20,* Adelaide, Australia, Australian Computer Society, Inc., 2003.

[10] Buck, D. and Stucki, D. J. Design Early Considered Harmful: Graduated Exposure to Complexity and Structure Based on Levels of Cognitive Development. *31st SIGCSE Technical Symposium on Computer Science Education,* 2000, 75-79.

[11] Buck, D. and Stucki, D.J. JKarelRobot: A case study in supporting levels of cognitive development in the computer science curriculum. *Proceedings of the 32nd SIGCSE Symposium on Computer Science Education,* ACM Press, New York, NY, 2001, 16-20.

[12] Buckley, J. and Exton, C. A framework for assessing programmers' knowledge of software systems. *Proc. 11th IEEE International Workshop on Program Comprehension, IWPC,* 2003.

[13] Burgess, G.A. Introduction to programming: blooming in America. *J. Comput. Small Coll.* 21, 19-28. 2005.

[14] Computing Accreditation Commission. *Criteria for Accrediting Computing Programs: Effective for Evaluations During the 2006-2007 Accreditation Cycle.* ABET Inc, Baltimore, MD, 2005.

[15] Cooper, S., Cassel, L., Moskal, B., and Cunningham, S. Outcomes-based computer science education *Proceedings of the 36th SIGCSE technical symposium on Computer science education,* ACM Press, St. Louis, Missouri, USA, 2005.

[16] Cukierman, D. and McGee Thompson, D. Learning Strategies Sessions within the Classroom in Computing Science University Courses*Proceedings of WCCCE 2007, 12th Western Canadian Conference on Computing Education,* May 2007.

[17] Doran, Michael V. and Langan, David D. A cognitive-based approach to introductory computer science courses: lesson learned. *Proceedings of the twenty-sixth SIGCSE technical symposium on Computer science education,* Nashville, Tennessee, United States, ACM Press, 1995.

[18] Facione, P. A. Critical thinking; A statement of expert consensus for purposes of educational assessment and instruction, research findings and recommendations, 1990*, Fullerton ERIC Reports*, ED315.423.

[19] Farthing, D. W., Jones, D. M. and McPhee, D. Permutational multiple-choice questions: an objective and efficient alternative to essay-type examination questions. *Proceedings of the 3rd Conference on Innovation and Technology for Computer Science Education, ITiCSE, 1998,* ACM Press, New York, NY, 1998, 81-85.

[20] Gronlund, N.F. *Measurement and evaluation in teaching.* MacMillan, New York, 1981.

[21] Hernán-Losada, I., Lázaro-Carrascosa, C. and Velázquez-Iturbide, J. Á. On the use of Bloom's taxonomy as a basis to design educational software on programming. *Proceedings of World Conference on Engineering and Technology Education, WCETE 2004,* COPEC, Brazil, 2004, 351-355.

[22] Hernán-Losada, I., Velázquez-Iturbide, J. Á and y Lázaro-Carrascosa, C. A. Programming learning tools based on Bloom's taxonomy: proposal and accomplishments. *Proc. VIII International Symposium of Computers in Education (SIIE 2006),* León, España, Octubre 2006, 2006, 325-334.

[23] Howard, Richard A., Carver, Curtis A. and Lane, William D. Felder's learning styles, Bloom's taxonomy, and the Kolb learning cycle: tying it all together in the CS2 course. *Proceedings of the twenty-seventh SIGCSE technical symposium on Computer science education,* Philadelphia, Pennsylvania, United States, ACM Press, 1996.

[24] Huitt, W. and Hummel, J. 2003. Piaget's theory of cognitive development. *Educational Psychology Interactive*

[25] Ihantola, P., Karavirta, V., Korhonen, A. and Nikander, J. Taxonomy of effortless creation of algorithm visualizations. *Proceedings of the 2005 International Workshop on Computing Education Research, ICER '05,* Seattle, WA, October 01-02, 2005, ACM Press, New York, NY, 2005, 123-133.

[26] Illinois Online Network: Educational Resources, http://www.ion.illinois.edu/resources/tutorials/assessment/bloomtest.asp, Accessed on 19/07/2007, 2007.

[27] Johnson, C. G. and Fuller, U. D. Is Bloom's taxonomy appropriate for computer science? *6th Baltic Sea Conference on Computing Education Koli Calling 2006,* Koli Calling, November 2006, Berglund, A. and Wiggberg, M., Eds. Department of Information Technology, University of Uppsala, Stockholm, 2007, 120-123.

[28] Joint IEEE Computer Society/ACM Task Force on Computing Curricula. 2005. The Overview Report. http://www.computer.org/portal/cms_docs_ieeecs/ieeecs/education/cc2001/CC2005-March06Final.pdf, 2005, visited September 2007.

[29] King, O.M. and Kitchener, K.S. 1994. *Developing reflective judgement: understanding and promoting intellectual growth and critical thinking in adolescents and adults.* Jossy-Bass Inc, San Francisco.

[30] Kolb, D. *Experiential Learning: Experience as the Source of Learning and Development.* Prentice-Hall, New York, NY, 1984.

[31] Kramer, J. Is abstraction the key to computing? *Communications of the ACM* 50, 37-42, 2007.

[32] Krathwohl, D.R., Bloom, B.S. and Masia, B.B. 1964. *Taxonomy of educational objectives: the classification of*

educational goals. Handbook Volume 2: Affective domain. McKay, New York.

[33] Krathwohl, D.R. A revision of Bloom's taxonomy: an overview. *Theory into Practice* 41, 212-218, 2002.

[34] Kumar, A.N. Learning programming by solving problems. In *Informatics Curricula and Teaching Methods,* L. Cassel and R.A. REIS, Eds. Kluwer Academic, 29-39, 2003.

[35] Kundratova, M., Turek, I. *Chapters from engineering pedagogy. Educational Objectives* (in Slovak). STU Bratislava, 2001.

[36] Lahtinen, E. and Ahoniemi, T. Visualizations to Support Programming on Different Levels of Cognitive Development. *Proceedings of The Fifth Koli Calling Conference on Computer Science Education,* 2005, 87-94.

[37] Lahtinen, E. A Categorization of Novice Programmers: A Cluster Analysis Study. *Proceedings of the 19th annual Workshop of the Psychology of Programming Interest Group,* Joensuu, Finland, July 2-6, 2007, Sajaniemi,J. and Tukiainen,M., Eds. University of Joensuu Department of Computer Science and Statistics, Joensuu, Finland, 2007, 32-41.

[38] Lister, R. On Blooming First Year Programming, and its Blooming Assessment. *Proceedings of the Australasian Conference on Computing Education*ACM Press, New York, NY, 2000, 158-162.

[39] Lister, R., and Leaney, J. Introductory programming, criterion-referencing, and Bloom. *Proceedings of the 34th SIGCSE technical symposium on Computer science education,* Reno, Nevada, USA, ACM Press, 2003.

[40] Lister, R., and Leaney, J. First year programming: Let all the flowers bloom. *5th Australasian Computer Education Conference,* Adelaide, SA, Australia, 2003.

[41] Lister, R., Adams, E.S., Fitzgerald, S., Fone, W., Hamer, J, Lindholm, M., McCartney, R., Moström, J.E., Sanders, K., Seppälä, O., Simon, B., and Thomas, L. A multi-national study of reading and tracing skills in novice programmers. *Working group reports from ITiCSE on Innovation and technology in computer science education,* Leeds, United Kingdom, ACM Press, 2004, 119-150.

[42] Lister, R., Simon, B., Thompson, E., and Whalley, J.L. Not seeing the forest for the trees: novice programmers and the SOLO taxonomy. *Proceedings of the 11th annual SIGCSE conference on Innovation and technology in computer science education,* Bologna, Italy, ACM Press, New York, NY, 2006, 118-122.

[43] Machanick, P. Experience of applying Bloom's Taxonomy in three courses. *Proc. Southern African Computer Lecturers' Association Conference,* Strand, South Africa, June 2000, 2000, 135-144.

[44] Manaris, B. and McCauley, R. Incorporating HCI into the undergraduate curriculum: Bloom's taxonomy meets the CC'01 curricular guidelines. *Frontiers in Education, 2004. FIE 34th Annual Meeting,* 2004, T2H/10-T2H/15.

[45] Merrill, M.D. Lesson segments based on component display theory. In *Instructional design theory,* M.D. Merrill, Ed. Educational Technology Publications, Englewood Cliffs, NJ, 177-212, 1994.

[46] Merrill, M.D. The prescriptive component display theory. In *Instructional design theory,* M.D. Merrill, Ed. Educational Technology Publications, Englewood Cliffs, NJ, 159-176, 1994.

[47] Merrill, M.D. The descriptive component display theory. In *Instructional design theory,* M.D. Merrill, Ed. Educational Technology Publications, Englewood Cliffs, NJ, 111-157, 1004

[48] Moon, J. *How to use level descriptors.* Southern England Consortium for Credit Accumulation and Transfer, 2002.

[49] Naps, T., Cooper, S., Koldehofe, B., Roessling, G., Dann, W., Korhonen, A., Malmi, L., Rantakokko, J., Ross, R.J., Anderson, J., Fleischer, R., Kuittinen, M. and McNally, M. 2003. Evaluating the educational impact of visualization. *ACM SIGCSE Bulletin* 35, 124-136.

[50] Niemierko, B. *Pomiar sprawdzajacy w dydaktyce. Teoria i zastosowania (in Polish).*Panstwowe Wydawnictwo Naukowe, Warszawa, 1990.

[51] Oliver, D., Dobele, T., Greber, M., and Roberts, T. This course has a Bloom Rating of 3.9. *Proceedings of the sixth conference on Australasian computing education - Volume 30,* Dunedin, New Zealand, Australian Computer Society, Inc., 2004.

[52] Perry, W.G.J. *Forms of intellectual and ethical development in the college years: a scheme.* Harcourt Brace Jovanovich College Publishers, Forth Worth, 1968.

[53] Perry, W.G.J. Different worlds in the same classroom. In *Improving learning: new perspectives,* P. Ramsden, Ed. Kogan Page; Nichols Pub. Co, London, New York NY, 145-161, 1988.

[54] Piaget, J. and Inhelder, B. *The Psychology of the Child.* Routledge & Kegan Paul, 1969.

[55] Polanyi, M. 1958. *Personal knowledge: towards a post-critical philosophy.* Routledge and Kegan Paul, Chicago.

[56] Rademacher, R. Applying Bloom's taxonomy of cognition to knowledge management systems. *1999 ACM SIGCPR conference on Computer Personnel Research,* New Orleans, LA, April 8-10, 1999, ACM Press, New York, NY, 1999, 276-278.

[57] Rapaport, W.J. William Perry's scheme of intellectual and ethical development, http://www.cse.buffalo.edu/~rapaport/perry.positions.html.

[58] Reeves, M.F. An Application of Bloom's Taxonomy to the Teaching of Business Ethics. *Journal of Business Ethics* 9, 609-616, 1990.

[59] Reigeluth, C.M. and Stein, F.S. 1983. The elaboration theory of instruction. In *Instructional-design theories and models: an overview of their current status,* C.M. Reigeluth, Ed. Lawrence Erlbaum Associates, Hillsdale, NJ, 338-381.

[60] Reigeluth, C.M., Merrill, M.D. and Bunderson, C.V. 1994. The structure of subject matter content and its instructional design implications. In *Instructional design theory,* M.D.

Merrill, Ed. Educational Technology Publications, Englewood Cliffs, NJ, 59-77.

[61] Reigeluth, C.M., Merrill, M.D., Wilson, B.G. and Spiller, R.T. 1994. The elaboration theory and instruction: a model for sequencing and synthesizing instruction. In *Instructional design theory,* M.D. Merrill, Ed. Educational Technology Publications, Englewood Cliffs, NJ, 79-102.

[62] Reynolds, C. and Fox, C. 1996. Requirements for a computer science curriculum emphasizing information technology: subject area curriculum issues. *ACM SIGCSE Bulletin* 28, 247-251.

[63] Robins, A., Rountree, J. and Rountree, N. 2003. Learning and Teaching Programming: a Review and Discussion. *Computer Science Education* 13, 137-172.

[64] Sanders, I. and Mueller, C. A fundamentals-Based Curriculum for First Year Computer Science. *31st SIGCSE Technical Symposium on Computer Science Education,* ACM Press, 2000, 227-231.

[65] Scott, T. Bloom's taxonomy applied to testing in computer science classes. *J. Comput. Small Coll.* 19, 267-274, 2003.

[66] Simpson, B.J. The classification of educational objectives: psychomotor domain. *Illinois Journal of Home Economics* 10, 110-114, 1966.

[67] Svec, S. Taxonomy for Teaching: A System for Teaching Objectives, Learning Activities and Assessment Tasks (Revision of Bloom's Taxonomy of the Cognitive Domain). In *Pedagogicka revue* (in Slovak) 57, 453-476, 2005.

[68] Swedish Ministry of Higher Education and Research, Higher Education Ordinance, http://www.sweden.gov.se/sb/d/574/a/21541, Accessed on 19/07/2007, 2007.

[69] Thompson, E. Does the sum of the parts equal the whole? *Proceedings of the seventeenth annual conference of the National Advisory Committee on Computing Qualifications,* Mann, S. and Clear, T., Eds. National Advisory Committee on Computing Qualifications, 2004, 440-445.

[70] Thompson, E. Holistic assessment criteria - applying SOLO to programming projects. *Proceedings of the Ninth Australasian Computing Education Conference (ACE 2007),* Ballarat, Victoria, Australia, Mann, S. and Simon, Eds. Australian Computer Society Inc, 2007, 155-162.

[71] University of Victoria. Learning Skills Program - Bloom's Taxonomy, http://www.coun.uvic.ca/learn/program/hndouts/bloom.html, Accessed on 19/07/2007, 2007.

[72] Whalley, J.L., Lister, R., Thompson, E., Clear, T., Robbins, P., Kumar, P. K.A., and Prasad, C. An Australasian study of reading and comprehension skills in novice programmers, using the bloom and SOLO taxonomies. *Proceedings of the 8th Australasian Conference on Computing Education - Volume 52,* Hobart, Australia, Australian Computer Society, Inc, 2006, 243-252.

[73] Winslow, L.E. 1996. Programming Pedagogy - a Psychological Overview. *SIGCSE Bull.* 28, 17-22.

Acknowledgements

This paper is partly supported by the ITiCSE 2007 Student Bursary of the British Computer Society and the grant of FGU No 21/2007 (Faculty of Management Science and Informatics, University of Zilina)