AC 2010-484: PROBLEM-BASE LEARNING OF MULTI-CORE PROGRAMMING

Wei Zhang, Southern Illinois University, Carbondale

Problem-Base Learning of Multicore Programming

Department of Electrical and Computer Engineering
Southern Illinois University Carbondale
Carbondale, IL 62901

1. Introduction

The computer industry is rapidly moving towards the multicore processors. Manycore processors have been widely used in all computing domains, including desktop, server and embedded computing systems. A multicore processor combines two or more independent processors into a single package, which is capable of executing multiple threads simultaneously. The L2 cache on a multicore processor can be either private or shared, as depicted in Figure 1 (a) and (b), respectively. Clearly, multicore processors can naturally benefit multithreaded programs by running them on different cores concurrently to improve the throughput. However, unlike other advances of microprocessors aiming at the transparent increase of single-threaded performance (e.g., frequency scaling, pipelines, caches, and superscalar architectures), multicore processors cannot automatically reduce the latency of single-threaded programs. In many cases, there is no way to effectively utilize the performance of additional processor cores or hardware threads without writing explicitly multithreaded programs [1]. In other words, the advent of multicore processors necessitates a fundamental shift of software development paradigm, from the traditional single-threaded programming to multithreaded programming, which is a great challenge.

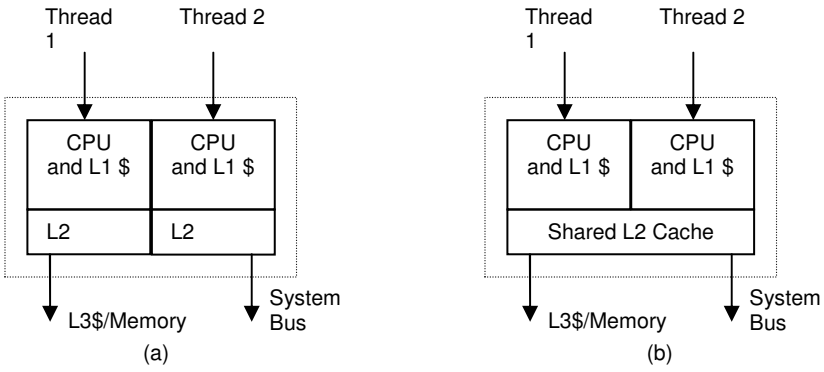

Figure 1. A dual-core processor with (a) private L2 caches or (b) shared L2 cache.

To prepare students for the coming era of multicore computing, I propose to develop a multicore programming course for our senior and graduate students. The goals of this project are to:

- Create a course on multicore programming for general-purpose applications, which can be potentially adapted or adopted by the computer engineering or computer science departments at other institutions.
- Investigate the effective use of problem-based learning (PBL) in this course to overcome the difficulty of teaching and learning MTP. More specifically, this project is expected to provide useful insights into several key PBL problems,

including: 1) how to select appropriate programming problems to ensure the breadth of contents covered, 2) how to balance teaching and students' self-directed study in programming courses, and 3) how to enhance the guided PBL model based on both qualitative and quantitative evaluation to improve students' MTP programming skills.
- Assess the effectiveness of developed PBL-based multicore programming course for students with diverse background.

As one of the few earliest courses on multicore programming for general-purpose computers, this course is expected to benefit students in our institution and beyond. This project is partially supported by the National Science Foundation (NSF). The developed course materials will be made freely available by the end of the project, which are expected to benefit more students and instructors that are interested in multicore programming.

2. Related Work

2.1 Prior Efforts in Teaching Multithreaded Programming

Historically, creating multithreaded software has mainly been the realm of developers focused on high-performance applications, such as demanding scientific applications or customized enterprise applications that require advanced programming techniques [2]. While traditionally there have been many courses on parallel processing [3, 4] and distributed computing [5-7], these courses typically target high-performance applications or web-based applications, which normally use specific languages (e.g. HPF, Split-C, Java), specific libraries (e.g LAPACK, Multipol), and specific parallel or distributed machines (e.g. Thinking Machines, Internet). In contrast, multicore processors are essentially single computers, which have been increasingly used in desktops and servers for general-purpose applications. However, students are usually not trained to develop MT programs for general-purpose applications. Moreover, programming for multicore processors needs to make different tradeoffs for maximizing performance, since different cores are on the same chip and can provide significantly larger inter-core communication bandwidth as well as shorter inter-core communication latency than parallel or distributed computers [8]. Consequently, in multicore software development, the concurrency extraction, thread partition, synchronization, and architecture-specific optimizations are quite different from those in traditional multithreaded programming for parallel and distributed computers.

Typically, MTP related topics such as the concept of threads, mutual exclusion, and deadlock, are covered in an operating system course within a few weeks. To facilitate student's learning of general MTP, Shene and Carr developed a pedagogical tool – ThreadMentor [9], as well as a comprehensive MTP course [10], which, however, does not cover multicore specific programming and optimization issues.
While it is already difficult for students to learn to write a "correct" MT program [10], developing "good" MT applications for multicore platforms will be even more challenging. To the best of our knowledge, only a few institutions, including MIT [11]

and Georgia Tech [12], have recently begun to offer multicore programming courses. However, both these courses [11, 12] focus on studying and using a particular multicore architecture, i.e., Cell processor [13] used in the Sony PlayStation 3, which may fundamentally limit the wide adoption of these courses by other institutions intending to educate their students in MT software development for general-purpose multicore processors. Also, both these courses [11, 12], to the best of our knowledge, do not study how to deal with students' difficulty in learning multicore programming. Moreover, these courses, as well as other aforementioned MTP, parallel and distributed programming courses, are taught by traditional lecture-based approaches, not by the problem-based learning approach.

Compared with prior work in multithreaded programming education, this project attempts to make the following unique contributions: 1) I will investigate the novel use of PBL to address the difficulty of teaching and learning multithreaded programming; 2) I will teach software development for general-purpose multicore based computers, which can be easily adopted by other institutions; 3) I will use authentic MTP problems, a professional development environment, and real dual-core processors (i.e. Intel Core 2 quad-core) throughout this course to enhance students' capability to solve "real-world" MTP problems; and 4) I will evaluate the effectiveness of PBL in enhancing students' learning outcome of multicore programming, and provide useful guidance for other instructors to enhance and apply PBL to multicore or other programming courses.

2.2 Brief Review of PBL

PBL is an instructional and learning method based on using problems as a starting point for acquisition and integration of new knowledge [14]. PBL embodies most of the principles that we know to improve learning, such as active learning, cooperating, getting prompt feedback, tailored to student's learning preference with student empowerment and accountability [15]. PBL was first applied in the 1960s, by the faculty of Health Sciences of McMaster University, Canada, and in the school of Medicine of Case Western University [16]. Since then, the PBL approach has flourished in many medical and professional schools. PBL has also been effectively used in some fields of STEM to improve students' learning, such as the PBL activities at the University of Delaware [17], the PBL lab at Stanford [18], the Samford PBL initiative [19], and the learning initiative at Penn State [20].

Recently, PBL has also been used in the fields of computer engineering and computer science. For example, Iowa State exploited PBL to revitalize a sophomore level embedded system course [21]. Kay and Kummerfeld created a PBL interface design and programming course [22]. Delaney and Mitchell used PBL in software engineering projects [23]. Garcia-Famoso applied PBL to an introductory course of computer architecture [16]. While all these prior efforts can provide valuable experiences for this project; to the best of our knowledge, there is no previous work to explore and evaluate PBL in addressing the difficulty of teaching and learning MTP for multicore processors.

3. Why PBL?

I propose to use PBL for reducing the difficulty of teaching and learning multicore programming, and for improving students' learning outcome. PBL is promising in this regard due to the following reasons:

1. Recent cognitive psychology found that "a student is not only an empty vessel waiting to be filled with new knowledge" [24]. Instead, learning is an active process of acquiring knowledge, and PBL is a pedagogical approach consistent with this finding. In particular, one major difficulty of learning MTP is that MTP requires a new mindset, since it is rather difficult for students to apply what they have learned and used for years (in single-threaded programming model) [4]. I believe that PBL will enable students to actively learn the new MTP model from the scratch, and thus can help them make this transition smoothly.
2. Programming is not only a science, but also an art [25]. Excellent programming skills cannot be simply learned by studying factual knowledge as emphasized in traditional lecture-based teaching. In a PBL-based course, students' learning is driven by the problems themselves, rather than by presentation of the subject content [26]. Thus PBL offers an excellent opportunity for students to practice, apply, and develop skills such as problem solving, reasoning, and self-learning [23], which are critical for students to develop excellent multicore programs.
3. Students have been found lack of early fostering of communication and collaboration skills that are seen by industry as paramount and essential for team based software development [27]. With PBL, students will work as a team, which mirrors the professional behavior of software developers. As a result, students' teamwork and leadership skills will be developed or improved.
4. Life-long learning is a necessity for software developers due to the rapidly and continually changing nature of the IT industry. PBL can develop students' self-learning capability, which can help them keep abreast of multicore computing and other new technology.

4. Apply PBL to the Multicore Programming Course

I plan to apply the guided problem-based learning [28] in the proposed multicore programming course. By comparison to a full PBL model, in which students are fully self-directed in their learning of new knowledge; in the guided PBL model, lectures would be used to present (in the problem context) fundamental concepts at appropriately-timed points in the problem development, and a range of resources would be available to provide assistance in detailed knowledge acquisition during the learning/interaction process [28]. I believe a guided PBL model, rather than a full PBL model, will lower the risk of introducing PBL into the multicore programming course while enhancing students' learning outcome.

I plan to teach the proposed PBL course based on a 4-step learning cycle, which will be evaluated and enhanced throughout this project. As depicted in Figure 2, firstly the instructor will introduce the real-world problem (or sub-problems) in detail and give

mini-lectures that last 10 to 15 minutes. Then the students will discuss in groups[1] to comprehend the problem, to develop a plan to acquire the necessary knowledge, and to propose possible solutions. Each group will then execute the plan during and outside of class time. To facilitate this self-learning process, students will be provided with additional lectures slides, surveys on particular topics, references, and useful web links. During the in-class discussions, the instructor will circulate among the groups as an unobtrusive "consultant". On return from self-learning, students will then re-analyze their hypothetic solutions, revise or improve the solutions through interactive discussions and the feedback from the instructor. This learning cycle will continue until the students have solved the given problem (or sub-problem) excellently. Then, the instructor can move on to the next sub-problem, and repeat the student-centered cyclic learning model.

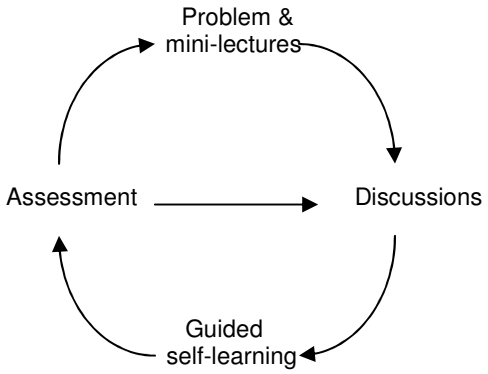

Figure 2. The 4-step learning cycle.

5. Course Development Efforts

So far I have created a new course entitled "ECE 432: Programming for Multicore Systems" at SIUC and have developed multicore programming course materials.

The Problem: To apply PBL in this course, a good multicore programming problem must be used to cover the main course objectives. The problem I select is the grep parallelization problem, as described below.

"The grep is a popular single-threaded command line utility originally used in the Unix operating system in 1970, which provides the capability to search generalized regular expressions occurring in Unix files. Multi-soft Inc., a startup company that aims at providing high-performance software tools for multicore processors, would like to develop a multithreaded search tool – mgrep, which can take advantage of both the inherent parallelism in string searching and the great potential of multicore processors. Compared with single-threaded grep, mgrep should provide noticeable performance improvement for large files. The project currently targets the Intel Core 2 quad-core processor. To be competitive in the software market, the goal of this project is to maximize the performance of mgrep on the Intel Core 2 quad-core computers in 4 months (i.e., a semester). In addition to the high performance, mgrep is also required to support

[1] To ensure enough time for discussions, this course will consist of a set of 3-hour classes meeting once a week.

both fixed pattern search and fuzzy string search with command line or user-friendly GUI (Graphical User Interface). Suppose you are a software engineer at Multi-soft Inc., working with a group of 4 other software engineers on this project, how will your group design and implement this project within the deadline?"

I have implemented a multithreaded version of grep and test it on multicore processors with different number of cores. A test file was created to run the initial test. The file was about 1Gbyte in size and it contained 15 unique expressions for which to search for. I observe significant performance improvement of mgrep as the number of threads and cores increase by using data level parallelism. Thus I believe it can potentially be a good project with an appropriate level of difficulty for students to explore.

It is important to note that the selected problem requires students to study a broad range of key MTP topics, e.g., multicore architecture, thread parallelization, and performance evaluation, which are included in the major educational objectives. Thus, the breadth of contents covered by this course is likely to be ensured, which will be guided by the instructor during the project development and be evaluated in the comprehensive test by the end of the course.

Course Modules: In order to help students manage and plan their work and to facilitate easy adoption and reusability, I divide this course into 10 modules, which are organized around 4 themes as described below.

Theme1: Multicore Architecture

- Module 1 – Introduction to the Intel Core 2 Quad-Core and generally Multicore Architectures: Technology trends and challenges of future microprocessor design; architectural and microarchitectural features of Intel Core 2 quad-core processor; comparison of single-core, hyper-threading, multi-processors, and multicore architectures; and the differences between multicore processors and parallel/distributed computers.

Theme2: Find Concurrency and Extract Threads

- Module 2 – Fundamentals about Threads: The meaning of threads, differences between processes and threads, creating and destroying threads, thread execution models, thread state diagram, thread priority and scheduling, kernel vs. user threads.
- Module 3 – Finding Concurrency: Introduction of task and data decomposition, dependence analysis, concurrent program and data structuring patterns.
- Module 4 – Multithreaded Programming Models: Pthreads or OpenMP, introduction of automatic multithreading by parallel compilation, and the tradeoffs among different approaches.

Theme3: Ensure the Correctness of Multithreaded Programs

- Module 5 – Synchronization: Race conditions, critical sections, reader/writer locks, spin locks, semaphores, and condition variables.

- Module 6 – Software Tools for Debugging Multithreaded Applications: Tutorials on the advanced graphical debugger (dbx) from Sun Studio 12, and GDB/DDD for thread examination.

Theme4: Maximize Multithreaded Performance on Multicore Chips

- Module 7 – Performance Measurement and Load Balance: Throughput vs. latency, Amdahl's law, mapping software concurrency to the parallel hardware, code transformation, data layout optimization, load balance, and thread migration.
- Module 8 – Software Tools for Maximizing Performance on Multicore Processors: Sun Studio performance analyzer, and utilities to profile programs, i.e., prof, gprof.

Labs: The following labs have been designed for the proposed course, which will be used in conjunction with the main project:

1. Performance comparison of single-threaded and multi-threaded programs {One 3-hour session}
2. Parallelize a for loop {Two 3-hour sessions}
3. Implement thread priority and scheduling {Two 3-hour sessions}
4. Implement synchronization on multithreads {Two 3-hour sessions}
5. Debug race conditions {One 3-hour session}
6. Performance tuning on a multicore processor {One 3-hour session}

Course Learning Outcomes / Expected Performance Criteria:

Upon completion of the course, the students should be able to:

- Understand the architecture of multicore processors
- Understand how to use Pthreads or OpenMP to develop multi-threaded programs
- Understand how to ensure the correctness of multithreaded programs
- Understand basic tools of tuning the performance of multi-threaded programs
- Understand and implement multithreaded code
- Design a multithreaded grep utility program for a multicore processor.

6. Evaluation

This course has been offered for the first time in Spring 2010 at the Electrical and Computer Engineering Department of Southern Illinois University Carbondale. At the time of writing this paper, it is the 8th week of this semester and I have just conducted a mid-term course evaluation and a few interviews with students. Although certainly more results can be reported by the end of the semester, I have already observed some interesting feedbacks from students taking this course.

In the evaluation form I handed out to students, students were asked to answer the following question "Do you prefer 1) traditional lecture-based teaching, 2) pure PBL approach, or 3) the guided PBL method as I am using in this course?" I have received 13 anonymous forms back to me. Students' answers to this question are shown in Table 1.

Students' Preference	Number of Students	Percentage of Students
Pure lecture-based teaching	1	7.7%
Pure PBL	2	15.4%
Guided PBL with lectures	10	76.9%

Table 1. Distribution of students' preference to pure lecture-based teaching, pure PBL and guided PBL.

As can be seen from Table 1, a majority of students, i.e. 76.9% prefer the guided PBL model, which combines PBL with traditional lecture-based teaching. These results confirm my hypothesis that applying pure PBL in this programming course may be too risky, and the guided PBL model can not only better motivate students to learn, but also ensure the full coverage of the knowledge. Since only 1 student prefers traditional lecture-based teaching, it seems that most students are demanding some degree of self-study and active learning through PBL, as pure lecture-based teaching tends to make students' learning process passive. This conclusion is also confirmed by the feedbacks from the two students who prefer pure PBL, as both of they mentioned that PBL helps to "motivate the student to study" or "require more thought before implementation".

One potential problem of pure PBL is that it may demand significant time and self-study from students, which seems to be a major reason students hesitate about PBL. Actually, the student who prefers pure lecture-based teaching commented that "I don't like having to spend time learning material on my own to complete assignments, with an already busy schedule". Therefore, it appears that the guided PBL model is preferable to the pure PBL model when teaching a single course, unless the whole curriculum is restructured based on PBL as many medical schools have done, which is out of scope of this paper.

All the students who filled out the evaluation forms indicated that developing multithreaded programs for multicore processors is a challenging topic for them. The major difficulties include "parallel thinking", "learning how to solve problems in parallel", and "synchronization", all of which are new problems students usually don't need to deal with in programming for traditional uniprocessors. Most of these students also mentioned that PBL help them to learn multicore programming. For example, one student commented that "PBL helps in making me think more about the problem and derive my own solution". Another student reported that "Problem-based learning helps me understand how everything I learn is connected". Based on students' feedback, it seems that using the guided PBL can help students study multicore programming.

An important feature of PBL is that students usually work as groups to solve given problems, which can not only help them learn from each other, but also improve their teamwork and communication skills. This is confirmed in my teaching of this course. Most students actually mentioned in their evaluation form that they prefer group work, because "it is easier to learn off each other and makes projects less overwhelming", and "we get to work with more ideas and the coding workload is distributed". However, group work is definitely not a "one-size-fits-all" approach. Two out of thirteen students wrote they prefer working individually rather than as a group. The reasons are that they

think “working as a group has the drawback that most of the time I am only solving part of the lab” and “I like to be exposed to each part of the lab or project”. Therefore, it would best serve all the students if the instructor gives students an option to either form a group or work individually if he or she prefers.

Also, we have just finished a mid-term exam in this class. The average score of this test is 84.5, which seems to be very good as compared to a typical 400-level ECE course. Certainly, no definite conclusion can be made regarding students’ performance in this case as different exams perhaps with different levels of difficulty are used in different courses. I plan to teach the same course without using the PBL based approach next time, and then more meaningful results may be got by comparing students’ performance in the same course with different teaching methods

7. Concluding Remarks

The multicore platform has gained wide acceptance by the microprocessor industry. However, programming for multicore processors is a very challenging task, because traditionally students are only trained in single-threaded programming for single-core systems, which cannot take full advantage of the parallel processing capability of multicore platforms. To address the challenge of developing parallel programs for multicore processors, I have developed and been teaching a new course on multicore programming. Compared to a few earliest efforts on teaching multicore programming in other universities such as MIT and Georgia Tech, I propose to apply a guided problem-based learning method to help students make smooth transition from the single-threaded programming model to the multithreaded programming model. Also, instead of focusing on specialized multicore architectures, this course targets a general-purpose multicore platform based on the Intel Core 2 architecture.

This new course has been offered in spring 2010 for the first time and I have already observed some interesting results by the middle of this course. It seems that most students prefer the guided PBL model than either the pure PBL or the traditional lecture-based teaching. Also, most students think that using PBL help them to learn multithreaded programming for multicore processors, which they all agree is a very challenging topic. In addition, students’ feedbacks indicate that working in a group in this course is helpful for most of the students; but two students still prefer to work individually to solve the whole problem rather than parts of the problem. Therefore, it seems the instructor should give students an option to either form a group or work individually to achieve the best learning outcomes for all students.

This project is an ongoing effort that is supported by the NSF CCLI program. Based on students’ performance in the labs, project, homework and mid-term exam, it seems that students’ learning outcome and satisfaction of this course are very good. While some interesting results have already been observed so far, more comprehensive conclusions can be made by the end of this semester or by the end of the project period. We hope the information and observation shared in this paper can benefit those who are interested in developing or adopting courses on multicore programming or are interested

in applying PBL in their teaching of computer engineering courses. The website for this new course can be found at http://www.engr.siu.edu/~zhang/ece432/.

References:

[1] H. Boehm. Threads cannot be implemented as a library. In Proc. of the 2005 ACM SIGPLAN conference on Programming language design and implementation, June 12-15, 2005.
[2] D. Kretsch. Beware: Concurrent applications are closer than you think. Sun Technical Article, December, 2005.
[3] B. Parhami, Introduction to parallel processing: algorithms and architectures. Plenum Press, New York, 1999.
[4] D. E. Culler, J. P. Singh and A. Gupta. Parallel computer architecture: a hardware/software approach. Morgan-Kaufmann Publishers, 1998.
[5] D. C. Hyde. An undergraduate distributed computing course. The 1999 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'99), June 28 - July 1, 1999.
[6] G. Coulouris, J. Dollimore and T. Kindberg. Distributed systems: concepts and design. Addison-Wesley, second edition, 1994.
[7] J. Farley. Java: distributed computing. O'Reilly and Associates, 1998.
[8] S. P. Amarasinghe. Multicores from the compiler's perspective: a blessing or a curse? Keynote Speech, International Symposium on Code Generation and Optimization (CGO), San Jose, CA, March 2005.
[9] S. Carr, J. Mayo and C-K Shene. ThreadMentor: a pedagogical tool for multithreaded programming. In ACM Journal on Educational Resourses in Computing, Vol. 3, Issue 1, March 2003.
[10] C. Shene and S. Carr. The Design of a multithreaded programming course and its accompanying software tools. The Journal of Computing in Small Colleges, Vol. 14 (1998), No. 1 (November), pp. 12 - 24.
[11] Homepage of MIT 6.189 Multicore Programming Primer: Learn and Compete in Programming the PLAYSTATION®3 Cell Processor.
http://cag.csail.mit.edu/ps3/index.shtml, April. 2008.
[12] Homepage of GIT ECE4893A/CS4803 Multicore and GPU Programming for Video Games.
http://users.ece.gatech.edu/~leehs/ECE4893/, April 2008.
[13] M. Gschwind. Chip multiprocessing and the Cell Broadband Engine. Invited Paper and Keynote Speech, Computing Frontiers 2006, May 2006.
[14] H. S. Barrows. Problem-based learning: an approach to medical education. Springer series on Medical Education, New York, 1980.
[15] S. Carr, J. Mayo and C-K Shene. ThreadMentor: a pedagogical tool for multithreaded programming. In ACM Journal on Educational Resourses in Computing, Vol. 3, Issue 1, March 2003.
[16] M. Garcia-Famoso. Problem-based learning: a case study in computer science. In Recent Research Developments in Learning Technologies, 2005.
[17] Homepage of problem-based learning at the University of Delaware.
http://www.udel.edu/pbl/courses.html.
[18] Homepage of PBL lab, Stanford. http://pbl.stanford.edu/.
[19] Homepage of the Samford PBL Initiative, Samford University.
[20] Hompage of Learning Initiative of the Penn State College of Information Sciences and Technology.http://pbl.ist.psu.edu/.
[21] A. Striegel and D. T. Rover. Problem-based learning in an introductory computer engineering course. In Proc. of the 32nd ASEE/IEEE Frontiers in Education Conference, November, 2002.
[22] J. Kay and B. Kummerfeld. A problem-based interface design and programming course. In Proc. of SIGSCE, 1998.
[23] D. Delaney and G. G. Mitchell. PBL applied to software engineering group projects. In Proc. of International Conference on Information and Communication in Education, 2002.
[24] J. C. Perrenet, P. Bouhuijs and J. Smits. The suitability of problem-based learning for engineering education: theory and practice. In Teaching in Higher Education, Vol. 5, No. 3, 2000.
[25] D. E. Knuth. Computer programming as an art. Communication of the ACM, Vol 17, Number 12, December 1974.
[26] A. Ellis et al. Resources, tools, and techniques for problem based learning in computing. In Annual Joint Conference Integrating Technology into Computer Science Education archive Working Group reports

of the 3rd annual SIGCSE/SIGCUE ITiCSE conference on integrating technology into computer science education, 1998.
[27] L. Ma, J. D. Ferguson, M. Roper, J. Wilson, and M. Wood. A collaborative approach to learning programming: a hybrid learning model. In the 6th Annual Higher Education Academy Subject Network for Information and Computer Science Conference, York, UK, August 2005.
[28] A. Ellis et al. Resources, tools, and techniques for problem based learning in computing. In Annual Joint Conference Integrating Technology into Computer Science Education archive Working Group reports of the 3rd annual SIGCSE/SIGCUE ITiCSE conference on integrating technology into computer science education, 1998.