Analysis Of The Interoperability Of Intelligent Tutoring Systems For Programming: An Aspect Of Programming Exercises

Bui Trong Hieu

Abstract: University introductory programming courses are part of the curricula of many engineering and sciences programs. These courses provide a set of programming exercises for students. However, for many novice students, it is very difficult to learn. In particular, these students often get stuck and frustrated when attempting to solve programming exercises. One way to assist beginning programmers to overcome difficulties in learning to program is to use intelligent tutoring systems (ITSs) for programming, which can provide students with personalized hints of students’ solving process in programming exercises. Many ITSs for programming that offer programming exercises provide automated feedback on student programs. In spite of the proven effectiveness of ITSs for programming, these systems are utilized in real classrooms. Because of interoperability issues, ITSs for programming are difficult to build in current educational platforms without additional work. This disadvantage is significant because ITSs for programming require considerable time and resources for their implementation. In this research work, the paper presents both a soft and technical interoperability of programming exercises. With regard to the soft interoperability of programming exercises, the paper discusses on a classification of programming exercises. With regard to the technical interoperability of programming exercises, this paper presents a framework that addresses the content and communication interoperability issues typically found in the domain of programming.

Index Terms: Classification, Intelligent Tutoring Systems, Interoperability, Programming Assignments, Programming Exercises Programming Tutors.

1 INTRODUCTION

PROgramming skills are becoming a core competency for almost every profession and thus, computer science education is being integrated in the curriculum for almost every study subject [1]. However, many students find great difficulty with the learning of programming and it becomes a barrier to their further studies of computer science and other disciplines. This difficulty is in large part due to students’ inability to solve their programming exercises, and this may discourage them to progress further when help can be obtained immediately. In order to address this problem, various approaches have been proposed to help students learn solving programming exercises. Traditionally, face-to-face and one-to-one human tutoring had been the best option for tutor. However, human tutors are not always available and that’s why computer based tutoring is developed to provide as an alternative support. Intelligent Tutoring System (ITS) is an example of computer-based tutoring which is developed emulating the human tutor [2]. According to VanLehn [3], an Intelligent Tutoring System that is designed with the ability to understand the coding to a low level of granularity in its advice can be just as effective as human tutor. ITSs for programming are useful particularly for first year computer science students and non-major students [4]. Many ITSs for programming that offer programming exercises provide automated feedback on student solutions. The aim of this paper is to 1) discuss on a classification of programming exercises and 2) present a framework that addresses the content and communication interoperability issues typically found in the domain of programming. The benefits of this classification are twofold. First, designers of ITSs for programming can choose or develop an appropriate modeling technique for a specific class of programming exercises.

Second, this classification serves researchers of the ITSs for programming community with a communication means to talk about types of programming exercises more precisely when they intend to exchange their programming exercises, e.g. for evaluation purposes. Based on this framework a network of ITSs for programming and services was created and deployed for the computer programming domain. This framework instance comprises several systems and services and their integration poses interoperability issues on two levels: content and communication. Content issues are tackled with a standard format to describe programming exercises as learning objects. Communication is achieved with the extension of existing specifications for the interoperation with several systems typically found in an e-learning environment such as ITSs for programming, learning objects repositories and online judge systems. Some of these systems were created from scratch and others were adapted to support the new content and communication specifications.

2 BACKGROUND

2.1 Intelligent Tutoring Systems

As noted above, face-to-face and one-to-one human tutoring is the best tutoring field. However, it is extremely expensive in terms of both physical and human resources. ITSs are a natural solution that can be used to address this problem, as they are developed to give personalized feedback and help to students who are working on problems. The fact the ITSs are formed by three fields: Computer Science, Psychology, and Education, as illustrated in Fig. 1, in which, (i) Artificial Intelligence (AI) addresses how to reason about intelligence and thus learning, (ii) Psychology (Cognitive Science) addresses how people think and learn, and (iii) Education focuses on how to best support teaching/learning [5].
According to Lee & Chen [6], an Intelligent Tutoring System (ITS) is a computer system that provides immediate and customized instruction or feedback to learners. The classical architecture of an Intelligent Tutoring System includes the following four components (Fig. 2) [7], [8], [9], [10], [11].

This traditional view of ITSs is still very accepted by the ITS community. However, recent studies stress functionality over structure [12], [13], [14], [15], [16], describing ITSs as having two main loops [12]: 1) the inner loop and 2) the outer loop (Fig. 3) [13]. The inner loop is responsible for providing personalized feedback, hints, and direct problem solving assistance to students. The inner loop also assesses students’ competence and registers it on the student model. Using the information that is obtained about the student, the outer loop performs task selection. Until tutoring is complete, repeat: tutor selects a task; until task is complete, repeat: tutor may present a hint; tutor updates the student model; tutor presents feedback on the step; } Fig. 3. ITS Loops. This research work is inspired from VanLehn’s [12] two loop characterization of tutoring systems. The main task of the outer loop is to select an appropriate programming exercise for the student. The inner loop is responsible for giving hints on student steps. Here, this research work focus on the inner loop. This work does not support an outer loop which can create an overall student model and intelligently choose which programming exercises to show to the student. Nesbit and colleagues [17], in their paper, “Work in Progress: Intelligent Tutoring Systems in Computer Science and Software Engineering Education”, research on ITSs has accelerated over the last decade, and scholarly interest in such systems has never been greater. ITS have been developed for a wide range of subject domains (e.g., mathematics, physics, biology, medicine, reading, languages, philosophy, information technology and computer science) and for students in primary, secondary and postsecondary levels of education. Founded on several decades of research on human cognition and intelligence, Intelligent Tutoring System is now a fast growing area in academia and industry. We now turn our attention to some cutting-edge research on Intelligent Tutoring System in a specific learning domain: programming [18].

2.2 Intelligent Tutoring Systems for Programming

In the past four decades, a variety of ITSs for programming have been built to provide tutoring services for programming problems. When it comes to functionalities, in general, ITSs for programming can be classified into five types: 1) curriculum sequencing, which constructs for each student an individual learning path, including individual selection of topics to learn, examples, and exercises; 2) intelligent analysis of student’s solutions, which focuses more on debugging and error diagnosis for complete student’s program; 3) program debugging support, which helps students learn to analyze programs; 4) interactive code-writing problem solving support, which provides students with personalized assistance in each code-writing problem solving step and 5) example based code-writing problem solving support which suggests the most relevant cases or examples to students. In the context of ITSs for programming, for brevity, this paper will use the term “ITSs for code-writing” to describe to the ITSs for programming for interactive code-writing problem solving support.

3. RELATED WORK

3.1 Classification of Programming Exercises

With respect to learning outcomes and assessment results, taxonomies of educational objectives are used worldwide to describe learning outcomes and assessment results, reflecting a student learning stage. A very well-known taxonomy is the original Bloom’s taxonomy. It is a classification of the different objectives and skills that educators set for students. Skills in the cognitive domain are divided into six levels in the order starting from the lowest order processes to the highest:
Knowledge, Comprehension, Application, Analysis, Synthesis and Evaluation [19]. According to Santos et al. [19], the SOLO (Structure of the Observed Learning Outcome) taxonomy makes no reference to the learner performance cognitive characteristics or to the affective dimension. It focuses on the content of the learner’s response to what is being assessed. It aims to identify the nature of that content and the structural relationships within it. The content could be designed to assess knowledge, cognitive skills or underlying values. The taxonomy can be used to establish the expected relationships between these different types of content. Le et al. [20] posed a classification of the degree of ill-definedness of educational problems based on the existence of solution strategies, the implementation variability for each solution strategy, and the verifiability of solutions. The classification divides educational problems into five classes: 1) one single solution, 2) one solution strategy with different implementation variants, 3) a known number of typical solution strategies, 4) a great variety of solution strategies beyond the anticipation of a teacher where solution correctness can be verified automatically, and 5) problems whose solution correctness cannot be verified automatically. Le and Pinkwart [21] proposed a classification of programming exercises supported in learning environments. They base their classification on the degree of ill-definedness of a programming exercise. Class 1 programming exercises have a single correct solution. Examples are quiz-like questions with a single solution, or slots in a program that need to be filled in to complete some task. Class 2 programming exercises can be solved by different implementation variants. Usually a program skeleton or other information that suggests the solution strategy is provided, but variations in the implementation are allowed. Finally, class 3 programming exercises can be solved by applying alternative solution algorithms. According to Ruf et al. [22], programming exercises can be classified by very different properties, such as the difficulty, the purpose, the type or form (e.g. open or multiple choice questions), or the context, to mention only some features. As noted by the authors, they captured what is given in the respective programming exercise and what the student has to do to solve the programming exercise. Then they stripped both criteria “given” and “to-do” from the context and formulated them in a generic way. Two programming exercises were classified in the same category if they have basically the same given and if the same is to do. More complex programming exercises, which involved more than one “to-do” were divided into corresponding parts and associated with multiple categories, i.e. an “atomic” programming exercise was made from each “to-do”, which was then used for further investigation. For each of these “atomic” programming exercises it can be distinguished between knowledge and cognitive process dimension comparable with learning objectives. Knowledge elements in tasks occur both in the given problem and in the expected solution, whereas the cognitive process hides in the description of what is to do (see Figure 4).

3.2 Interoperability of Programming Exercises

In early study, Pillay [23] proposed a generic architecture for the development of ITSs for code-writing (see Figure 5). According to Pillay [23], in order to facilitate interoperability, each module is treated as an agent or team of agents which communicate using an agent communication language. In 2012, the interoperability issue of programming exercises has been further investigated by Queirós and Leal [24]. In their work, the authors identified three interoperability facets required for flexible assessment systems, namely easy configuration of new exercises, management of users, and report of assessment results i.e. marks and feedback. On the other hand, the conclusions of the survey highlighted the need for interoperability and propose their integration with e-learning systems, since these systems are ready for deployment, which most universities have them deployed, and already include interoperability features for users, grades and feedback. On year later he proposed BabeLO (see Figure 6) which is a programming exercise converter providing services to a network of heterogeneous e-learning systems such as contest management systems, programming exercise authoring tools, evaluation engines and repositories of learning objects. Its main feature is the use of a pivotal format to achieve greater extensibility. The BabeLO approach is a service to convert programming exercises formats on-the-fly. The BabeLO service is an open-source project and was already deployed in networks of e-learning systems that require on-the-fly conversion among learning object formats [24].
In their work, Gustavo Soares Santos and Joaquim Jorge [23], proposed an approach for the development of interoperable e-learning standards. In their approach, "atomic" tutoring systems are grouped to create "molecular" tree structures that cover course modules (see Figure Fig. 7). In this approach, the authors used SCORM compliance for developing.

In 2015, Amorim et al. [25] present an e-learning content selection model, based on multi-agent paradigm, aiming to facilitate the learning material reuse and adaptability on Learning Management Systems (LMSs). The proposed model was developed according to a (Belief, Desire, Intention) BDI multi-agent architecture, as an improvement of the Intelligent Learning Objects approach, allowing the dynamic selection of Learning Objects (LOs). The new proposed model is based on Multi-Agent System (MAS) approach integrated to a LMS, resulting on the improvement of the analyzed related works, allowing that LO can be included to the learning experience dynamically, adding intelligent behavior to the system. According to Amorim et al., to produce more intelligent LO, previous researches proposed the convergence between the LO and MAS technologies, called Intelligent Learning Objects (ILO). The objective of the new model called Intelligent Learning Object Multi-Agent System (ILOMAS) (see Figure Fig. 8) is to enhance the framework developed to create ILO based on MAS with BDI architecture, extending this model to allow the production of adaptive and reusable learning experiences.

In the context of Massive Open Online Courses (MOOCs), according to Staubitz et al. [26], dedicated development tools are supplied as tightly integrated parts of the MOOC platform. Tight integration can also be achieved if the development tool is only loosely coupled with the platform, e.g. by employing the Learning Tools Interoperability (LTI) standard for data exchange between MOOC platform, LMS and ITS. In recent 2018, Abdullah and Ali [26] provide a summary of standards used in the e-learning industry, and discussed the most cited...
organizations that are authorized to approve specifications as standards. Also it presents quality standards and their frameworks, and gives an overview on personalization and adaptation learning. Their research work also covers trends of e-learning standards. As noted by the authors, it is important to incorporate the approaches, techniques, tools, and trends in this branch of learning.

4 CONCLUSION

In the context of ITSs for programming, this paper presents both a soft and technical interoperability of programming exercises. With regard to the soft interoperability of programming exercises, the paper discusses on a classification of programming exercises. With regard to the technical interoperability of programming exercises, this paper presents a framework that addresses the content and communication interoperability issues typically found in the domain of programming.

REFERENCES


[22] Ruf, A., Berges, M., & Hubwieser, P. Classification of programming tasks according to required skills and knowledge representation. In International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, 2015, pp. 57-68.


