Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Interface creation and redesign techniques in collaborative learning scenarios

Benjamin Weyers[a,*], Wolfram Luther[a,1], Nelson Baloian[b,2]

[a] Department of Computer and Cognitive Science, University of Duisburg-Essen, Lotharstrasse 65, 47058 Duisburg, Germany

[b] Department of Computer Science, University of Chile, Santiago de Chile, Chile Avenida Blanco Encalada 2120, Santiago 6511224, Chile

ARTICLE INFO

Article history:
Received 30 October 2009
Received in revised form
23 March 2010
Accepted 7 May 2010
Available online 20 May 2010

Keywords:
Interface redesign
Concept keyboard
Cooperative learning
Distributed learning
Iterative interface creation

ABSTRACT

User interfaces are redesigned for various purposes, like adapting interfaces or meeting new requirements during software creation processes. In the context of learning systems, the aim of interface redesign is to let the student creates his or her own interface corresponding to the abstract concept to be learned, which is reflected in the interface designed. In this article we present an approach to interface redesign in a cooperative learning scenario for cryptographic protocols. We describe an iterative workflow using two different pieces of software for the creation and redesign of interfaces and distributed simulation and evaluate this approach.

1. Introduction

Computers have been widely used to support learning using problem-solving methodology, especially in collaborative environments [1,2]. This has often been called computer-supported problem-based Learning (PBL) [3]. Problems in this process may be either theoretical or practical. For theoretical problems, the PBL approach is used to model those problems and collaboratively discuss research questions dealing with various issues. For example, the security of cryptographic protocols can be modeled formally by students and then analyzed using various mathematical formalisms and verification tools in order to test them [4]. Modeling and testing environments can be done with computer-based tools. For example, there are tools to model Petri nets and general graphs and to simulate complex systems supporting interaction and the simultaneous exchange of ideas between learners. We should note that the problem-solving approach can be applied not only for learning but also for research purposes.

For practical, application-dependent problems, learners are typically asked to analyze a concrete issue in a collaborative manner. At the end of the process, they may or may not produce a solution. For this kind of problem, the computer usually provides the learners with tools to display the problem, solve some sub-problems, combine the sub-problems, share the proposed solutions with their co-learners, and, possibly, check the correctness of the solution through simulation.

The current literature shows PBL being applied to one or another of these types of approaches to learn complex algorithms through (a) formal and theoretical modeling or (b) application-dependent, practical simulation. However, there are problems that are better solved by considering them in both ways. Thus, PBL could benefit from expanding to include both the theoretical and the practical point of view. For example, let us consider the problem of determining the correctness of a certain algorithm and applying it to a concrete case. Because solving this problem as a whole provides a better understanding of the algorithm's inner workings, it has the advantage of applying the algorithm in the right way. The mutual enrichment of insights between the theoretical and the practical point of view in the context of PBL is worth investigating. We therefore propose developing a computer-supported learning methodology taking a holistic approach, that is, enabling the learner to explore a given problem by switching at will between its theoretical and its practical aspects.

The work reported in this paper seeks to create a learning environment with this holistic characteristic (cf. Fig. 1) by developing an activity that involves learners in designing and redesigning a software interface that implements a collaborative simulation of several cryptographic protocols. The goal of the learning activity is to design easy-to-use and intuitive interfaces for distributed software on two various layers, with each learner assuming the role of an actor in the protocol—sender, receiver, intruder, or certification agent. To create the interface, on a first layer (called the application layer; cf. Fig. 1), the students model the physical appearance of the interface that implements the practical

* Corresponding author. Tel.: +49 203 379 1224; fax: +49 203 379 3557.
E-mail addresses: weyers@inf.uni-due.de (B. Weyers), luther@inf.uni-due.de (W. Luther), nbaloian@dcc.uchile.cl (N. Baloian).
[1] Tel.: +49 203 379 3555; fax: +49 203 379 3557.
[2] Tel.: +56 2 978 4954; fax: +56 2 689 5531.

0167-739X/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2010.05.001

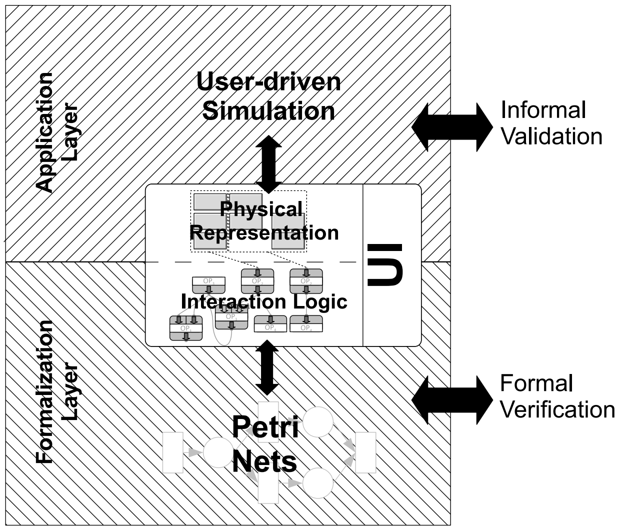

Fig. 1. General overview of the field of research.

perspective of the algorithm for simulation. Thus, every key is conceptually associated with a step in the cryptographic algorithm, representing its theoretical perspective. These associated concepts are explicitly modeled on a second theoretical layer (called the formalization layer; cf. Fig. 1) describing the logic of the interface, also called interaction logic. Through creating and reconfiguring an interface from both perspectives, students acquire insight to both layers. On a physical layer, the students organize the sequencing of the operation in the algorithm or group the buttons related to the role that has to execute them (sender, receiver, or certification agent). To create the theoretical layer, the students must know which concept/operation is associated with each button. In this way, they gain insight into the individual steps of the algorithm.

This concept is implemented using concept keyboards (CKs), which are physical or virtual keyboards that are easy to reconfigure and where each key triggers an atomic or complex action—in this case, an action that is part of the cryptographic protocol being studied. CKs have already been successfully used in various implementations and evaluations of learning systems for developing single-user scenarios supporting the learning of complex algorithms and concepts [5,6].

In addition to the creation of a CK, a verification step to identify errors in the modeled CK through cooperative simulation is also necessary. When errors occur, the students or the system initiates a reconfiguration of the interface. Therefore, creation and simulation are embedded in an iterative and cooperative workflow, starting with an introduction to the specific learning content (here, the cryptographic protocol) and finishing with an evaluation of the simulation process. For the implementation and evaluation of this approach, we introduce CoBo, a system for distributed simulation of cryptographic protocols, and UIEditor, a system for visual modeling of user interfaces.

The UIEditor software is used for a more general approach to interface creation and redesign. Its implementation in this context for the creation and redesign of CKs is completely new. It implements a visual modeling interface realizing both the physical and the logical layer of a user interface. The user can control the physical appearance of the CK, such as the number of buttons, their shape, size, and position, color, label, etc., with respect to the mental interaction model based on proven or newly conceived interaction patterns. The outward appearance is an important issue in CK creation and reconfiguration. The user has to establish a close relation between the physical interaction elements, such as buttons, and the concept the user associates with them. In other words, the user associates the outcome of a subpart of the algorithm or process and its actions with a specific button. This formal level of an interface can be modeled visually in a second part of the UIEditor. Thus, on the practical layer, the student describes his understanding of the algorithm concerning sequences of targeting operations in the physical representation of the CK, and, on the theoretical layer, he becomes involved in the concepts behind any button on the CK by modeling its action logic. The UIEditor can run on a touch screen computer to offer collaborative work for creation and reconfiguration of CKs.

CoBo is a system developed first in a student project in 2006 [7]. It offers a communication platform for distributed simulation of cryptographic algorithms based on formal models that also offer error recognition. The simulation is steered by the students through the use of a role-dependent CK that they have created using the UIEditor. Thus, every student represents one specific role in the protocol. The simulation offers a verification and validation of the created CKs and identifies possible errors on the theoretical layer. In this way, CoBo implements the combination of informal validation through visual simulation, algorithm animation and formal verification through matching the interaction logic created by the students to the model of the algorithm implemented by an expert. Based on this combination of these two ways to present the algorithm, its formal representation automatically created by the logical layer of the CK can be used for the simulation.

In addition to the developed workflow and the implemented software, we will show the benefit of this approach for the learning process in a case study as a result of our research. In an evaluation involving 66 students, we were able to show that cooperative creation of CKs supports the understanding of cryptographic protocols and enables the solving of more complex questions compared to a non-cooperative approach. Furthermore, we will show that this kind of cooperative work is highly motivating and that iterative creation of CKs is preferred compared to other well known learning approaches.

The next section describes the state of the art in relevant areas related to this work. Section 3 introduces the learning scenario and the developed system, including CoBo and UIEditor. Section 4 presents an evaluation of the workflow of the learning activity and the system itself, which was conducted in June 2009 at the University of Duisburg-Essen in the context of a course on algorithms and data structures. Section 5 concludes the paper and presents some ideas for future work.

2. Recent work

In the context of the work presented in the paper at hand, various areas of research are addressed and have to be discussed concerning recent work. The research results presented in this paper are concerned with the iterative and cooperative workflow 2.1 for learning cryptographic protocols, which addresses interface redesign techniques 2.2 based on concept keyboards 2.3 and algorithm visualization techniques 2.4 for validation, error recognition, and the initiation of interface reconfiguration.

2.1. Collaborative learning

Our main motivation for creating a cooperative workflow for learning complex cryptographic protocols is that collaborative learning has shown group work to have an important impact on achieving learning success [8–10]. Small group work presents opportunities for students to share insights [11] and observe the strategies of others [12], which is helpful in the context of algorithm learning. Publications such as [13] as well as work on Petri net-based collaborative design [14] are especially relevant to cooperative problem-solving systems for user interface design because of the need for a formal description of the interface for generating feedback for the students by verification.