Learning UI Functional Design Principles through Simulation with Feedback

Jenny Ruiz, Estefanía Serral, and Monique Snoeck, *Member, IEEE*

***Abstract*—The User interface (UI) is a key component of an interactive software application; therefore, it is important to provide software developers with basic UI design skills. However, teaching UI design is challenging, even at a basic level, and there is little teaching support. In this paper, we investigate the benefits of the feedback-enriched simulation environment (FENIkS) for learning fundamental UI design principles toward designing the functional aspects of a UI. FENIkS is a model-driven educational environment making use of simulation as learning support by generating a UI and the underlying application starting from conceptual domain and presentation models. The generated application and UI contain feedback elements for the learners. This feedback shows if the generated prototype is compliant with certain key design principles and why the principles are considered to be well applied or not. We conducted an experiment using bachelor-level courses to observe the effects of FENIkS on the learning of UI design principles by novice UI designers. The effectiveness of FENIkS was measured by comparing the results of students on a test performed without and with the use of FENIkS, using statistical methods. The findings show an improvement in student's learning of UI design principles when using the FENIkS approach.**

***Index Terms*—Automated feedback, computer science education, educational technology, user interfaces.**

I. Introduction

NOWADAYS, software applications are present in almost every aspect of our daily life. The user interface (UI) is considered one of the critical components of software applications because it enables users to interact with the software [1]. There is a growing interest in the study of UI design. Consequently, there is a need for skilled professionals capable of designing high-quality UIs. However, software engineering courses focus mostly on the programming, design, and architecture of the application logic while giving limited attention to the UI design. Often, software developers seldom receive further training on UI design; therefore, they should learn basic UI design principles. Although learning can be challenging, various UI design characteristics make the teaching of UI design difficult: task and tool complexity, rapid technological changes, and the inherent difficulties associated with teaching complex learning tasks [2].

The four-component instructional design (4C/ID) method for complex learning tasks [2] emphasizes the importance of training students based on "whole" and "real-life" tasks. Although "part-task practice" is essential in developing partial competences, learners must integrate multiple partial competences into a global competence to address complex tasks [2]. Therefore, the students need to be trained using "whole" tasks based on real-life tasks to ensure the transfer of knowledge to their future work environment.

Some tools have been developed to offer "part-task practice," including worked examples and games. Simulation has the advantage of being capable of addressing whole complex tasks and has been used in multiple domains as a teaching tool, including areas of knowledge like mathematics [3], [4], physics [5], and engineering [6]–[8]. During the development of learning tasks, unlike real-life tasks, simulation gives teachers more control over the complexity of the tasks, the amount of student support provided, and mitigation of the risks associated with failed task execution. Simulation is often applied in UI development by using tools such as mock-ups and wireframes. However, it rarely addresses "whole" tasks practice, as prototyping often focuses only on the UI. Furthermore, it lacks learner's support in the form of easy-to-use technology or automated in-tool feedback.

This research aims to improve the teaching of UI design by developing an educational simulation environment that enables "whole-task" practice while removing the difficulties associated with complex UI design tools. It requires collecting requirements for the learning environment and developing the environment. The benefits of using the educational simulation environment for students' achievement of learning goals are evaluated by conducting experiments, using an experimental design that controls unwanted variations while ensuring equal benefits for all students.

The remainder of this paper is structured as follows: Section II presents the problem statement, discusses related work, and outlines the general research approach. Section III gives an overview of FENIkS and highlights the learning benefits. Section IV presents the methodology for the experimental evaluation and describes the method for measuring the effects of the proposed simulation technique on the learning outcomes and shows the results. Section V discusses the limitations of the experimental assessment. Then, Section VI concludes the work.

This research has been sponsored by the VLIR-UOS network program. *(Corresponding author: Jenny Ruiz.)*

J. Ruiz is with the Faculty of Informatics and Mathematics, University of Holguin, 80100, Cuba (e-mail: jruizp@uho.edu.cu).

E. Serral and M. Snoeck are with the Faculty of Business and Economics, KU Leuven, 3000, Belgium (email: estefania.serralasensio@kuleuven.be; monique.snoeck@kuleuven.be).

II. Problem Statement, Related Work, and Proposed Research Approach

A. *Difficulties in Teaching UI Design*

Creating a UI is a complex task, and several factors contribute to making UI design difficult to teach, even at a basic level [9]–[11]:

The inherent complexity of the UI design: The field of UI design is inherently multidisciplinary; therefore, difficult to teach [12]. It involves several areas of computer science, such as computer architecture, operating systems, and computer graphics. Designers need to analyze the user-application interactions and model the interactions using various types of models, including task, user, and UI models. Thus, hands-on exercises require a considerable amount of knowledge from the students [13], which makes teaching UI design to students with low prior knowledge very difficult.

Rapid technology changes: Due to the rapid development of technology, there are various platforms and devices for which UIs can be designed. Developing UIs for multiple platforms, which provide the same functionality, further increases the difficulties in teaching UI design [14]. Therefore, students should be able to extrapolate knowledge and solve problems in new situations without taking into account technological details [15]. This requirement, however, increases the burden on teachers as they need to evolve their teaching tools.

The complexity of UI design tools: There are multiple UI design tools, which students need to learn. However, there is a potential risk of students focusing on understanding how to use the tools instead of learning how UIs should be designed.

The three earlier mentioned problems result in *general pedagogical difficulties of teaching complex tasks*. Complex learning tasks are characterized by the need to integrate various types of knowledge. Moreover, there are multiple valid ways to accomplishing tasks. These characteristics also apply to UI design [16], [17]. The integration of knowledge is stimulated by having students work on “whole” tasks instead of fragmented tasks. Notably, “active” learning is the better approach to complex learning [18], and practicing what has been learned facilitates teaching and is more rewarding for the students [19]. Solving these pedagogical problems is challenging in UI design. Providing students with feasible *“whole tasks”* and practice are made difficult by the inherent complexity of UI design, further exacerbated by tool complexity and rapid technological changes. The multiple valid solutions for a single design problem call for individual feedback, preferably immediate and explanatory rather than only corrective [20], resulting in an increased workload for teachers. As explained in [21], feedback on student performance is a central part of formative assessment approaches and improves students learning.

B. *Existing Tools and Approaches to Teaching UI Design*

1) *General tools for teaching UI design*

Within the UI design community, various efforts have been made to address the difficulties in teaching UI design. Notably, the difficulties associated with the complexity of tools have been addressed by providing tailored and simplified educational environments. The difficulties associated with the inherent complexity of UI design have been addressed by offering streamlined, simplified partial tasks to the students as opposed to real-life “whole” tasks. We discuss general tools and simulation tools to understand the success factors for teaching UI design.

Barrett [12] proposed a hypertext module that presents interface design principles and provides examples of good and bad interfaces, thus, simplifying the task and eliminating the need for students to interact with UI design tools. The tutorial includes the explanation of using metaphors, input devices, output methods, and evaluation issues. The knowledge is presented through textual descriptions, static and interactive examples.

The multimedia design advisor tool proposed in [22] is based on a simplification strategy. Multimedia designers can select appropriate media types for various information types using the tool. The tool works as a wizard giving recommendations on appropriate media based on the information type illustrated with examples.

Gamification is used in addition to simplification in the UsabilityGame [23]. The game shows a corporate environment to the player simulating real situations in the projects of a fictitious company. The game supports the teaching of the usability engineering life cycle, requirements analysis, prototyping, and heuristic evaluation. It shows a list of web interfaces with a heuristics list, where the student needs to select which of the heuristics have been applied.

ILIAS is proposed in [24] to address usability issues in an open-source learning management system. ILIAS users learn how to improve the use of ILIAS as a learning management system. The approach provides a taxonomy of UI components within ILIAS and guidelines for how to use them. It is an example of task simplification by offering a partial task and focusing on students’ familiar environment.

2) *UI simulation tools*

A few approaches support UI design using simulation based on the generation of prototypes. Prototypes can assist in addressing more complex and a larger variety of tasks, such as facilitate communication with the stakeholders, especially with the end-users [25], thus enabling students’ training in this aspect of the job. A simulation of interactive behavior with different adaptive menu algorithms using low-fidelity prototypes was presented in [26]. The simulation results are used to explore the possible outcomes of human performance for various menu generation algorithms. Simulated users are also used by [27] to test the probability that a user will experience difficulties when transitioning from one screen to another.

DynaMo-AID [28] generates a prototype from task and dialog models. Trætteberg [29] proposed a tool for the generation of prototypes from UI and domain models. This approach offers a concrete visual representation of the UI and enables prototype testing using an executable dialog model. Da Cruz and Faria [30] proposed an approach for generating a functional prototype from domain and use case models. GUILayout++ [31] enables the creation of prototypes with an abstract representation of UIs that can be used for evaluating the structure of the UI based on the composing areas. An automated layout approach for UI generation is proposed in [32]. This approach enables specifying layout parameters that can be used for rapid prototyping and initial user evaluation. CIAT-GUI [33] enables the visualization and modification of intermediate prototypes of the graphical UIs from domain and task models. Sottet et al. [10] generated prototypes from partial configurations using interaction flow and domain models. These tools for UI design simulation facilitate evaluating possible designs and can be useful teaching tools for this specific learning goal. However, none of them have been created for teaching UI design, and therefore they lack ways to support learners during their learning process. Notably, while using these tools, designers do not receive any instructional feedback that helps to learn UI design. The approaches that are similar to FENIkS are DynaMo-AID [28] and Trætteberg [29], which enable simulating a UI through the generation of prototypes from conceptual models. However, in both cases, the generated prototype is not completely functional, thus not enabling the training of students on "whole" tasks. Furthermore, these approaches are applicable in different contexts. DynaMo-AID addresses context-sensitive UI for mobile applications, while Trætteberg's approach addresses the generation of graphical UIs. Dynamo-AID cannot handle usability. Trætteberg's approach improves usability by integrating UI behavior and real UI components. However, none of the environments provide instructional feedback to the designer on how well design principles have been applied, and they are not designed for novice designers. The goal of FENIkS is to improve on the *"whole task"* and feedback aspect compared to existing simulation tools.

C. General Research Approach

The review of existing work reveals 1) that existing educational environments simplify the tasks substantially by starting from predefined examples of UIs; 2) that simulation tools are not enabled for "whole-task" training, and lack educational features such as feedback or recommendations. Prototyping has proved to be effective in evaluating different design alternatives [34]. Therefore, UI prototyping could be an effective way to learn UI design. In addition to the "whole-task" training, the feedback is important. Providing individual and immediate feedback is a key factor for skills acquisition [35], [36]. Previous research has demonstrated that simulation with integrated feedback positively contributes to the learning process [37], enabling the learner to "learn by experiencing" [38] and promoting successful transfer of knowledge to the real-world environment [39]. Experimental research on teaching UI design indicates that patterns and guidelines positively impact the novice designer's performance [40]; therefore, we propose to base the feedback on general UI design principles. Furthermore, difficulties associated with tool complexity should be resolved, and instructional design guidelines for complex learning should be observed. Consequently, the general research approach is 1) define requirements for the educational support for learning UI design; 2) develop an educational simulation tool; and 3) test the impact of applying the tool on the learning of UI design.

The contributions of this research are as follows:

1) We identify the requirements that an educational tool needs to provide support for teaching the functional aspects of UI design.
2) We develop an educational environment that satisfies those requirements, referred to as FENIkS.
3) We perform an empirical study that validates the teaching support of FENIkS for the functional aspects of UI design.

III. Simulation with Feedback in UI Design Teaching: FENIkS

In Section III-A, we describe the requirements to support the teaching of UI design. We describe FENIkS in Sections III-B and III-C. Section III-B describes the overall approach to generate a prototype. Section III-C describes how learning support is integrated into FENIkS. Subsequently, Section III-D analyzes the extent FENIkS satisfies the requirements described in Section III-A. A more extensive description of the technical design of FENIkS is in [41]–[43].

A. Requirements for Teaching Functional UI design

A teaching approach should satisfy the following requirements to facilitate the learning of the functional aspects of UI design:

R1: Outcome-oriented guidance. Multiple approaches can be followed to achieve a good UI; therefore, the focus should be on assessing the quality of the outcome instead of the quality of the approach. The quality of a UI design is achieved by observing good practices and principles that resulted from studying human behavior with computer systems [44]. Therefore, the educational environment should assist in observing these good practices of UI design.

R2: Immediate Feedback. In [45], the influence of feedback is addressed, and the essential components that make it effective are also defined. Providing individual and immediate feedback during learning is a key factor for skills acquisition [35], [38]. Task-level corrective and explanatory feedback informs learners and teachers how well tasks are performed and understood (and why) [20], and how close the learners are to accomplishing desired outcomes [46]. The educational environment should provide immediate corrective and explanatory feedback to the learners about their UI design.

R3: "whole-task" practice through the co-design of an application and its UI. Teaching UI design is complex because it depends on multiple types of knowledge and competences. Teachers typically address this using compartmentalization and fragmentation of a "whole task" into "part tasks" [18], for example, by teaching only the design of UIs and leaving out application development. However, in such a fragmented approach, students do not learn the full implications of the design decisions they made, particularly how UI design is closely related to the functionality of the application. Therefore, integrating UI design and application development can contribute significantly to the development of better systems [47] by explicitly establishing the link between the UI and the application logic.

R4: Ease of use. Designing UIs is a difficult process [11], [48]; therefore, ease of use is a critical factor [49]. The environment should be simple, understandable, flexible to interact with, and easy to become skillful at. Ease of use is important to avoid students focusing on learning the use of a tool rather than learning UI design.

R5: Learning by experiencing through simulation. UI design simulation allows trying different possibilities and visualizing the effects. Consequently, it enables learning from experience by allowing a student to explore what-if scenarios. Simulation with real-life cases is beneficial for complex learning [2], [18], and facilitates the transfer of knowledge to real-life environments [38], [50].

R4 is needed to address tool complexity, R1, 2, 3, and 5 are based on instructional design principles for complex learning [18].

B. Simulation with UI Generation in FENIkS

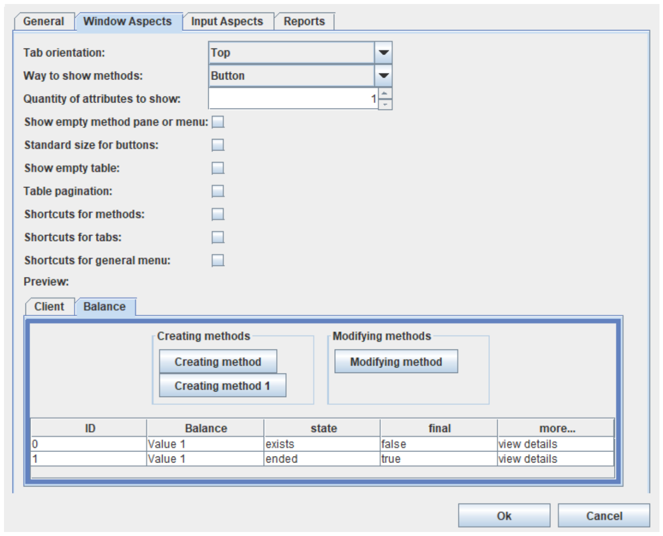
We extended an existing tool to create the proposed simulation environment. Particularly, FENIkS is based on JMermaid, a tool that has been used for more than 10 years for teaching conceptual modeling using simulation with feedback [51]. JMermaid was developed based on the concepts of MERODE, a model driven engineering (MDE) method [52] that enables specifying an enterprise's conceptual domain model that is platform-independent and sufficiently complete to automatically generate a prototype. The prototype contains instructional feedback [51], [53], [54], the effectiveness of which has been proven through several experimental evaluations [50], [54].

To support the design of a non-default UI, JMermaid was extended with two additional models and their corresponding generation of code: an Abstract User Interface (AUI) model and a Presentation Model (PM), this constituting the FENIkS approach [41].

The PM consists of three different views, and each is shown in a different tab in FENIkS: 1) the General aspects view, which collects the name of the application and other information to be shown in the title, 2) the Window aspects view, which captures preferences about how widgets will be shown (see Fig. 1), 3) the Input aspects view to capture the preferences related to input functionality of the application, such as how to configure input fields for attributes or how to show to-be-selected associated objects [42], [43] (see Fig. 2).

The tool provides default options; thus, designers do not need to specify values for all the design options. As a result, the generated code can be obtained at the early stages of the development process, provided the models are correct (meaning they need not be complete).

Fig. 3 shows the generation process. The process starts with the design of the platform-independent conceptual domain model and the PM. When the designer clicks the "generate" button, first, a model-to-model (M2M) transformation is executed to obtain the AUI model, which is also platform independent and is the expression of a UI in terms of interaction units without making any reference to the implementation. Second, model-to-code (M2C) transformations generate the prototype from the conceptual domain model and the AUI model.



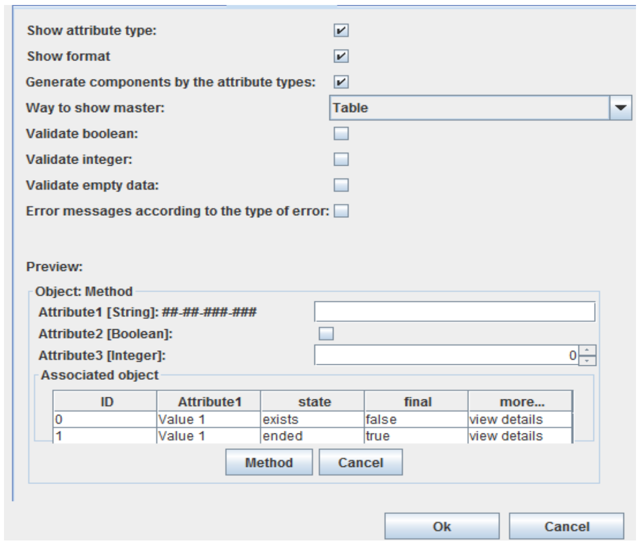
Tab orientation: Top
Way to show methods: Button
Quantity of attributes to show: 1
Show empty method pane or menu: ☐
Standard size for buttons: ☐
Show empty table: ☐
Table pagination: ☐
Shortcuts for methods: ☐
Shortcuts for tabs: ☐
Shortcuts for general menu: ☐
Preview:

Client | Balance

Creating methods: Creating method, Creating method 1
Modifying methods: Modifying method

ID	Balance	state	final	more...
0	Value 1	exists	false	view details
1	Value 1	ended	true	view details

Fig. 1. Window aspects of the PM in FENIkS.



Show attribute type: ✓
Show format ✓
Generate components by the attribute types: ✓
Way to show master: Table
Validate boolean: ☐
Validate integer: ☐
Validate empty data: ☐
Error messages according to the type of error: ☐

Preview:

Object: Method
Attribute1 [String]: ##-##-###-###
Attribute2 [Boolean]: ☐
Attribute3 [Integer]: 0
Associated object

ID	Attribute1	state	final	more...
0	Value 1	exists	false	view details
1	Value 1	ended	true	view details

Fig. 2. Input aspects of the PM in FENIkS.

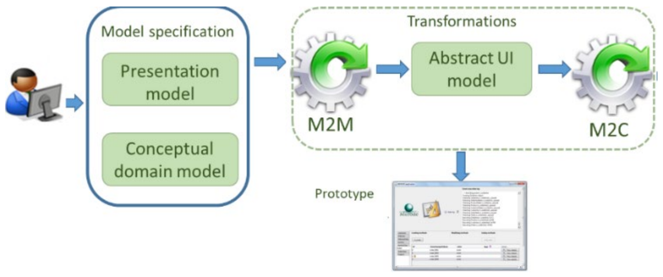


Fig. 3. Generation process with FENIkS.

The transformation steps are transparent to the designer, who only sees the final result, the application. The designer can regenerate another prototype by using the UI options stored in the PM. Changes in the UI options are shown on the UI accordingly. Furthermore, the application logic can be changed by updating the domain model.

C. *Feedback for Learning UI Design Principles*

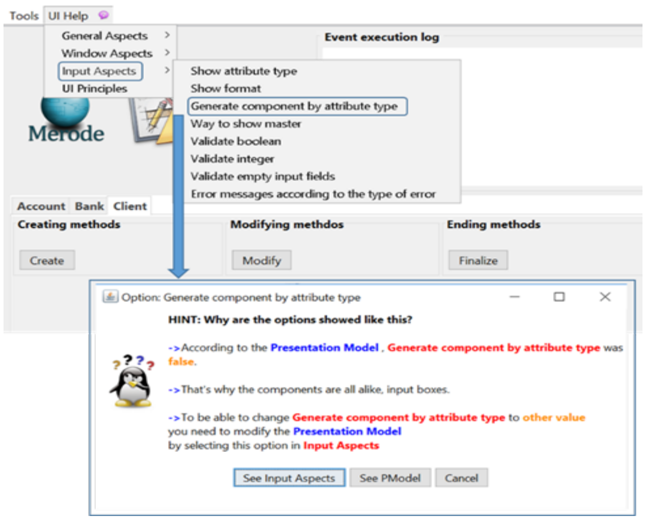
The design of FENIkS incorporates feedback to support the learning of UI design principles. FENIkS provides three types of instructional feedback related to the UI design: 1) immediate visual feedback using a preview; 2) explanatory feedback explaining why the UI is generated in a specific way and tracing the application's appearance to its origin in the PM, and 3) corrective feedback telling the learner whether the chosen design decisions are compliant with design principles.

Immediate visualization: At the bottom of the Windows and Input aspects views, the tool offers a preview of the to-be-generated UI that is automatically updated based on the selected UI options (see Fig. 1 and Fig. 2). The preview feature supports the learners by visualizing the generated UI and tracing changes in the model to their effects before generating the prototype.

In the generated prototype, explanatory and corrective feedback is provided as a "UI Help" feature in the application's main window. The designer selects General, Window, or Input aspects, and an option to see the explanation.

Fig. 4 shows an example of explanatory feedback corresponding to the Input aspects where the learner is provided with the feedback for the "Generate components by attribute type" option. The feedback is divided into three parts: 1) what the stored values are in the PM: 2) what the consequences are for the generated prototype, and 3) how to make modifications. From this view, it is possible to view the PM and the chosen values of the Window aspects.

Furthermore, FENIkS provides corrective feedback about compliance with two categories of UI design principles. First, there are four "*to actively observe*" principles, which compliance is influenced by the designer's choices in the presentation model. Depending on the chosen options in the presentation model, the principles are well applied or violated in the generated prototype.



Option: Generate component by attribute type

HINT: Why are the options showed like this?

->According to the Presentation Model , Generate component by attribute type was false.

->That's why the components are all alike, input boxes.

->To be able to change Generate component by attribute type to other value you need to modify the Presentation Model by selecting this option in Input Aspects

Fig. 4. Design options explanatory feedback.

The "*to actively observe*" principle's are:

- Allow users to use either the keyboard or mouse.
- Provide visual cues.
- Good error messages.
- Prevent errors.

Each of these principles has associated concrete guidelines for which options have been defined with two types of values: a positive value that makes the generated prototype compliant with the guideline and a negative value that implies guideline violation. The possible values are shown to the learners as UI design options in the presentation model. The chosen values are stored and used for the UI generation.

Second, three additional principles are "*observed by default*" in FENIkS, by which we mean that the generated prototype complies with these principles by default. The feedback explains the reasons why they are well applied:

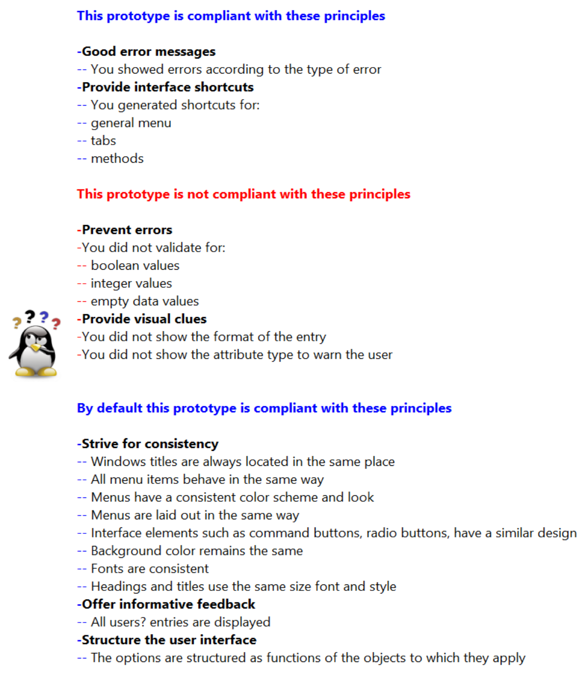
- Structure the UI.
- Strive for consistency.
- Offer informative feedback.

Using the values for the design options stored in the PM, FENIkS checks for each principle to be observed if the values selected by the learner are correct. Then it informs the learner on which principles the prototype is (partially) compliant with and which principles it is not to provide automatic feedback on principle compliance to the learner.

Fig. 5 shows an example of the feedback: besides the names of the principles, the feedback also explains the rationale behind the compliancy level and how the *"observed by default"* principles are satisfied.

The most important factors of the feedback provided by FENIkS, based on the framework presented in [55], are described as follows. Although the purpose of the provided feedback is corrective, it is also explanatory because it provides knowledge about the correct response (see Fig. 5). The feedback is accessible anytime, but the learner can also request this feedback before generating the prototype. It enables the designer to make the necessary changes in the PM to be compliant with all the principles, without needing to generate the prototype. Nevertheless, the generation is helpful for the complete visualization of the outcomes of choices in the PM.

All feedback in the tool is formative: It provides learners with information and guidelines to improve their answers while performing the learning task. As explained in [45], formative feedback is useful to students to improve their understanding level. The feedback is provided at Task-level and addresses how well tasks (in this case design principles) are understood, performed, or applied. It focuses on faults in the interpretation of the principles and the outcome produced. The recipient of the feedback is an individual learner, and the educational setting is the university. Learner control is achieved by letting the student decide when and where to see the feedback in FENIkS or the prototype (see Fig. 5).



This prototype is compliant with these principles

-**Good error messages**
-- You showed errors according to the type of error
-**Provide interface shortcuts**
-- You generated shortcuts for:
-- general menu
-- tabs
-- methods

This prototype is not compliant with these principles

-**Prevent errors**
-You did not validate for:
-- boolean values
-- integer values
-- empty data values
-**Provide visual clues**
-You did not show the format of the entry
-You did not show the attribute type to warn the user

By default this prototype is compliant with these principles

-**Strive for consistency**
-- Windows titles are always located in the same place
-- All menu items behave in the same way
-- Menus have a consistent color scheme and look
-- Menus are laid out in the same way
-- Interface elements such as command buttons, radio buttons, have a similar design
-- Background color remains the same
-- Fonts are consistent
-- Headings and titles use the same size font and style
-**Offer informative feedback**
-- All users? entries are displayed
-**Structure the user interface**
-- The options are structured as functions of the objects to which they apply

Fig. 5. Checking the UI principles in the generated prototype.

Similar to DynaMo-AID [28] and Trætteberg's approach [29], FENIkS generates a prototype from conceptual domain models, which help the designer see how the application would look and function.

D. *Requirements Satisfaction*

FENIkS satisfies the requirements presented in Section III-A as follows:

R1: Evaluating the outcome against good practices of UI design is satisfied using the principles for which FENIkS incorporates various design options and feedback.

R2: Immediate Feedback is satisfied because students can ask for feedback any time before and after generating the prototype.

R3: "Whole-task*"* practice and integration of knowledge via the co-design of an application and its UI are satisfied because it is possible to "co-design" the conceptual domain model and the UI design models. The integration enables the management of consistency and links between all the models and easily switching between adapting the application or adapting the UI. Students can thus address all aspects of application design.

R4: Ease of use. Once the conceptual and PMs have been created, the prototype can be generated with only one click. This produces an "illusion of simplicity," which makes the tool easy to use. There is no need to have a perfect or complete set of models to test the UI and the application code. Furthermore, the prototype can also be generated at early stages from a minimal domain model consisting of only one object type. The default values of the PM can be used, without needing to specify them. The possibility of generating the prototype from incomplete models enables checking partial versions of the prototype faster and contributes to ease of use.

R5: Simulation. There are two types of simulation incorporated in FENIkS: simulation by generating a fully working prototype and simulation of UIs using the runtime preview. An important added value is that this preview has the same layout as the to-be-generated UI. It also supports combining several generation options at once, enabling developers to assess the result of the overall generation process, instead of on a per-option basis.

IV. Experimental Evaluation

In this section, we present the experimental study we conducted to assess the effectiveness of FENIkS. Specifically, to which extent FENIkS is adequate to support the learning of UI design principles by software engineering students who have no prior training in UI design. Therefore, such students are considered novice designers.

A. *Measurement Instruments*

The effectiveness of FENIkS to support the learning of UI design principles is measured by student's test scores on an assessment of their knowledge about UI design principles. Furthermore, perceived usefulness and satisfaction are measured to evaluate the acceptance of the environment by the students.

1) *Knowledge assessment*

Students were given a set of UIs of a system where some UI design principles were well applied, and other principles were violated to test the *knowledge of UI design princi*ples. The students had to answer a series of questions (Exercises A/B, see Appendix A) to assess learning outcomes at different cognitive levels of Bloom's taxonomy [56]. Next, we explain some illustrative examples (see the complete test in Appendix A):

Cognitive level "understand": in Exercise A, questions 3–8, 10, 11, 13, students need to explain why certain items are an example (or not) of a principle. For instance, question 3 presents several statements where the student has to indicate which of these is a correct application of the principle "Structure the UI."

Cognitive level "apply": in Exercise A, questions 1, 9, 12, students need to identify the correct application of a principle. For instance, question 9 presents different options for errors that appear in the system. The student has to select the correct message to comply with the principle "*Good error messages*".

Cognitive levels "analyze" and "evaluate": question 2 in Exercise A asks the student whether the way the system asks a user for information is correct and why.

As explained in Section III, FENIkS provides the novice designer with feedback about several UI design principles. During the experiment, the students answered questions about the four "*to actively observe*" design principles and the three "*observed by default*" *principles*. The student cannot manipulate the design to violate these principles in FENIkS (see Section III). Furthermore, questions about three additional principles for which there is no feedback were also added to compare the effect of using FENIkS on the learning of the principles for which it provides support: *Visibility, Minimize user's memory load, Speak the user language*.

2) *User acceptance of the learning environment*

Previous studies [57] suggested perceived usefulness as an important factor for contributing to user acceptance for computer-assisted learning environments. In this research, we used a questionnaire (see Appendix B) composed of 15 items based on the one proposed by [50]. The items have 7-point Likert scales, anchored at the endpoints with the terms "Strongly disagree" for 1 and "Strongly agree" for 7. Questionnaires make it easier for the students (who feel somewhat uncomfortable during class discussions) to give feedback to the researchers [58].

B. *Experimental Design and Variables*

The experimental variables and hypotheses used in this experimental study are as follows.

1) *Dependent variables*

The dependent variable is the learning of UI design principles, measured through the score on a set of questions. For each correct answer, 1 point was given, and 0 was given for each wrong answer. Forty points could be obtained in the experiment, including 4 points per tested UI design principles (10 principles tested).

2) *Independent variables and treatment*

The independent variable used in this study is the use of feedback-enriched simulation. The goal of the experiment was to use FENIkS (the treatment) to influence the dependent variable (the learning of UI design principles). In this study, we used the feedback in conjunction with the preview capabilities and further simulation to enable students to link the conceptual model and the UI options in the generated working prototype.

An important step in running experiments is conducting pilots. As explained by [59], "for novel tools it can be particularly important to detect usability problems with a tool's UI that could interfere with participants' ability to succeed in the tasks." Therefore, before conducting the experiment, as shown in Fig. 6, two pilot experiments were conducted. The first pilot experiment was conducted during the first semester of the 2015–2016 academic year with 12 novice developers to evaluate the tool from the perspective of perceived usability by novice designers by performing an experiment. No participant had prior knowledge of the tool. We used the Computer System Usability Questionnaire (CSUQ) [60]. Each participant was asked to carry out a set of tasks in FENIkS, and then fill the CSUQ. The scores for all the items ranked above 5 (on a 7-point scale), indicating a positive evaluation. Developers found FENIkS very satisfactory in all areas: usefulness, information quality, and interface quality. FENIkS is positively perceived overall and provides the functionalities the developers expected. Details of this pilot experiment are described in [42].

The second pilot experiment was conducted during the second semester of the 2015–2016 academic year, with 20 students enrolled in the 4th year of the Informatics Engineering program at the University of Holguin to check the feasibility and correct setup of the experiment. The pilot experiment started with teaching the UI design course to the mentioned group of students through lectures and without using FENIkS or any other tool. After the lectures, a first test was taken by the students. Subsequently, the students learned how to use FENIkS, followed by a second test, equivalent to the first one, but in this case, students were allowed to use FENIkS as a help to perform the test. The second pilot experiment led to the adoption of randomized control crossover design for the final experiment. We also improved the tests following the pilot experiment by adding more questions per principle.

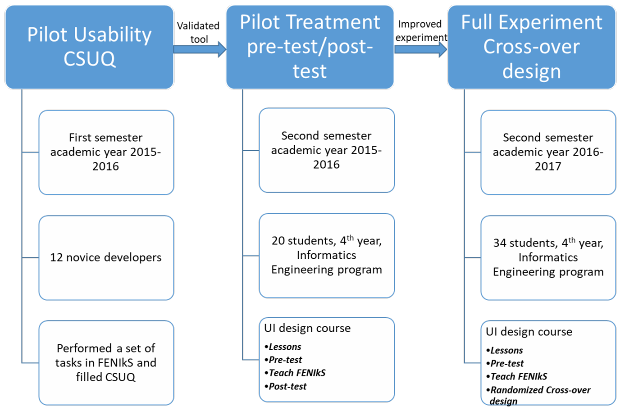

Pilot Usability CSUQ
Validated tool
Pilot Treatment pre-test/post-test
Improved experiment
Full Experiment Cross-over design
First semester academic year 2015-2016
12 novice developers
Performed a set of tasks in FENIkS and filled CSUQ
Second semester academic year 2015-2016
20 students, 4th year, Informatics Engineering program
UI design course
•Lessons
•Pre-test
•Teach FENIkS
•Post-test
Second semester academic year 2016-2017
34 students, 4th year, Informatics Engineering program
UI design course
•Lessons
•Pre-test
•Teach FENIkS
•Randomized Cross-over design

Fig. 6. Sequence of pilots and experiments.

C. *Experimental Details*

The final experiment, which is the focus of this paper, used a crossover design in which the treatment consisted of creating a UI using FENIkS. A crossover design was chosen to enable all students to use FENIkS while serving as their control group. The crossover design significantly reduces between-subject variability and permits between and within-group comparison [61]. Furthermore, it has the advantage of eliminating bias that may be introduced by the professor during the course delivery in the two sections (teaching without FENIkS and teaching the use of FENIkS) and eliminating any attitude bias that might result if students of either section received only the treatment or control for the entire course if swapping did not occur [62].

The experiment was performed in the 2016–2017 academic year with 34 students of the 4th year of the Informatics Engineering program at the University of Holguín (see demographic details in Table I).

TABLE I
DEMOGRAPHIC DATA

		Amount	%
Gender	Male	29	85.29%
	Female	5	14.71%
Age distribution	Min age	21	
	Max age	29	
	Average age	23.88	
	St. dev	2.38	

Fig. 7 shows the experimental setup and how it is embedded in the course. The learning of UI principles was tested three times. First, the students received lectures about UI design principles without using any tool. Subsequently, they participated in a pre-test to measure their knowledge of UI design principles. Then, the students learned how to use FENIkS and completed Exercises A/B on the same day. Students had one hour to complete each exercise, which was a paper-and-pencil test. Both exercises aimed to answer questions about specific design choices in UI design and whether the design choices are in line with UI design principles. The pre and post-test group experiments are based on a quasi-experimental design made up of two groups of 17 students in each group.

In line with the crossover design, students were randomly assigned to Group 1 or 2 and the treatment consisted of using FENIkS. Group 1 did Exercise A without using FENIkS and then Exercise B with FENIkS, while Group 2 did Exercise A with FENIkS and Exercise B without FENIkS. We ensured the exercises are equivalent to minimize differences in scores due to varying difficulty levels. (See Appendix A for details of the exercises.) In both exercises, the students were asked four questions per principle and were also asked to write a short motivation of the answers for a more accurate scoring of their competences. The summary is used to identify answers where students guessed the correct answer. Answers without a summary or with contradictory summary were scored as a wrong answer.

We collected the students' demographic and other information using questionnaires administered at the end of the experiment. We included a question on self-reported prior knowledge to verify the students' level of expertise in terms of UI design. Furthermore, a questionnaire on perceived usefulness and satisfaction was used to measure the acceptance of the learning environment, perceived usefulness, and satisfaction.

Crossover design may present some carryover effects. Carryover effects can be mitigated by extending the time gap between the treatment (the use of FENIkS) and the no-treatment experiments with the expectation that the carryover effect would disappear during the gap. However, as pointed out in [63], this strategy increases the total duration of the trial. Conversely, the presence of a carryover effect for Group 2 possibly shows that the students internalize the learning support provided by the tool. In previous research using a similar experimental design, the internalization of the support became apparent because students of Group 2 were using the terms of the learning tool when performing the exercises without the tool [64]. It shows an improvement in their ability to reason on models without tool support.

D. *Hypothesis*

Using the experimental design detailed above, we can answer the research question, "Does the use of FENIkS improve the novice designers' learning of principles related to functional aspects of UI design?" by testing the following hypotheses:

H1: Students make fewer errors with FENIkS than without FENIkS.

H2: When using FENIkS, students make fewer errors in UI design principles for which FENIkS provides support ("*to actively observe*" or "*observed by default*"), compared to when not using FENIkS.

Based on this hypothesis, it is expected that:

Students in Group 1 will make fewer errors in exercise B (solved with FENIkS) than in exercise A (solved without FENIkS). Specifically, students of Group 1 will make fewer errors against the "*to actively observe*" UI design principles in FENIkS in Exercise B than in Exercise A.

Students in Group 2 will make fewer errors in Exercise A (solved with FENIkS) than in Exercise B (solved without FENIkS). Specifically, students in Group 2 will make fewer errors against the "*to actively observe*" UI design principles in FENIkS in Exercise A than in Exercise B.

H3: When using FENIkS, students make fewer errors for the "*to actively observe"* UI design principles than those that are only observed by default or without teaching support.

Using the experimental setup, we can compare the results of the two groups, and it is expected that:

The improvement for the "*to actively observe*" principles in FENIkS is greater than for the "*observed by default*" principles.

The improvement for the "*to actively observe*" UI principles in FENIkS is greater than for the principles without teaching support.

H4: The use of FENIkS has a short-term persisting learning effect on student's test scores when it is no longer used.

The setup of the experiment is such that Group 1 will not use FENIkS for the first exercise. Conversely, Group 2 will not use FENIkS for the second exercise after completing the first exercise using FENIkS. Therefore, we will be able to compare the results of the two groups for Exercise A and Exercise B without using FENIkS, expecting that students from Group 2 will make fewer errors in Exercise B than students from Group 1 in Exercise A.

H5: FENIkS is suitable for novice UI designers (user acceptance).

This hypothesis is tested using the questionnaires that measure the perceived usefulness and satisfaction at the end of the experiment.

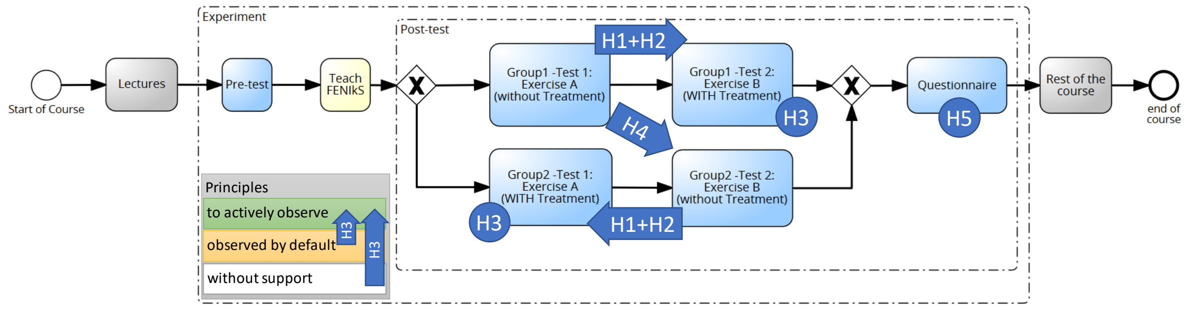


Fig. 7. Experimental setup.

E. Results of the Evaluation

Table II presents the results of the question on self-reported prior knowledge.

TABLE II
PREVIOUS KNOWLEDGE (SELF-REPORTED) OF UI DESIGN

		The experiment
Previous knowledge (self-reported)	No knowledge	12.12%
	Little knowledge	18.18%
	Moderate knowledge	51.52%
	Extensive knowledge	18.18%

We measured how many errors the students made after completing both tests.

1) Analysis of the error rates

Table III shows the pre-test results, which measure the students' knowledge of UI design principles (Min score 0, Max score 10). The average score and distribution of students above and below average for both groups are similar.

TABLE III
PRE-EXPERIMENTAL TESTING FOR BASIC KNOWLEDGE

	$\overline{X}_{\text{error}}$	Average (<5)	Average (≥5)
Group 1	4.41	9	8
Group 2	4.35	10	7

The data of the exercises with and without the treatment were analyzed using a statistical comparison of error rates with paired and two-sample t-tests [65]. Assuming a normal distribution of the differences between the errors before and after the treatment, we can perform a paired t-test. Appendix D shows the distribution of the differences in errors.

We tested the normality of the differences in errors using the Shapiro Wilk normality test [66], which is recommended for testing the normality of data [67]. The *p*-value for the entire group (0.799) is greater than 0.05. Furthermore, the *p*-value for Group 1 and 2 is 0.440 and 0.921, respectively. Therefore, the values appear to be normally distributed based on the results of the normality test.

Table IV shows the paired t-test to compare mean error rates (without and with FENIkS) for the entire group, and Groups 1 and 2 separately. We observe a decrease in errors with the use of FENIkS, as shown in Table IV. Therefore, we accepted hypothesis H1: Students make fewer errors with FENIkS than without FENIkS.

TABLE IV
MEAN ERROR RATES: T-TEST, PAIRED TWO SAMPLE FOR MEANS

	$\overline{X}_{\text{error without}}$	$\overline{X}_{\text{error with}}$	$\overline{X}_{\text{difference}}$	*p*-value one-tailed	*p*-value two-tailed
Entire group	15.94	13.15	−2.79	0.000***	0.000***
Group 1	16.06	13.12	−2.94	0.000***	0.000***
Group 2	15.82	13.18	−2.64	0.005**	0.010*

Cohen's *d* = 0.87 shows that the effect size is large, which is a significant result. Both the one-tailed and two-tailed *p*-values are very low ($p < 0.01$, 99% confidence interval). The effect size is large for Group 1 with Cohen's *d* = 1.12 and medium for Group 2 with Cohen's *d* = 0.71.

2) Analysis of different kinds of principles

For hypotheses H2 and H3, we need to analyze the differences between the principles, based on the type of support provided by the tool. Notably, we compared the average improvements for "*to actively observe*" and "*observed by default"* principles without teaching support (H2), and the improvement of "*to actively observe*" over the "*observed by default*" principles (H3). Table V shows the t-test results for various principles.

TABLE V
WITHIN-GROUP ANALYSIS: PAIRED T-TEST FOR ERRORS

			$\overline{X}_{\text{error without}}$	$\overline{X}_{\text{error with}}$	$\overline{X}_{\text{difference}}$	*p*-value one-tail	*p*-value two-tail
Principles with teaching support	*"To actively observe" principles*	Group 1	5.76	3.76	−2	0.000***	0.000***
		Group 2	5.24	3.7	−1.54	0.006**	0.012*
	"Observed by default" principles	Group 1	4.41	4.88	0.47	0.135	0.271
		Group 2	4.71	3.65	−1.06	0.023*	0.046*
Principles without teaching support'		Group 1	5.88	4.47	−1.41	0.001**	0.001**
		Group 2	5.88	5.82	−0.06	0.468	0.936

Comparing the improvement for principles without teaching support with the improvement for principles with teaching support, we observe that the students performed better for principles with teaching support with significant results in all the tests except for Group 1 for the "*observed by default*" principles. Notably, comparing the ones without teaching support and the "*to actively observe*" principles, we observe that the improvement for the latter principles is more significant, as shown in Table V. Two-tailed for Group 1 "*to actively observe*" vs. principles without teaching support: 0.000*** < 0.001**; two-tail for Group 2 "*to actively observe*" vs. principles without teaching support: 0.012* < 0.936. The results support H2: when using FENIkS, students make fewer errors in UI design principles with teaching support in FENIkS than for UI design principles for which FENIkS does not give feedback.

A paired t-test shows a significant improvement for both student groups for "*to actively observe*" principles in FENIkS. The effect size is large for Group 1 with Cohen's *d* = 1.21 and medium for Group 2 with Cohen's *d* = 0.69. For "*observed by default*" principles, there is no improvement for Group 1, but there is an improvement for Group 2. However, the effect size is small for both groups with Cohen's *d* = 0.28 and 0.49 for Groups 1 and 2, respectively. The results support H3, with the improvement for the "*to actively observe*" UI principles higher than for the "*observed by default.*" For the principles without teaching support, Group 1 has a difference, but no difference for Group 2. The effect size for Group 1 is large, with Cohen's *d* = 0.94; conversely, the effect size for Group 2 is small, with Cohen's *d* = 0.02.

To gain more insight into the support of the hypotheses H2 and H3, we calculated the error rates for each principle. The comparison of mean error rates between principles with and without support in FENIkS is presented in Table VII. Generally, for all the principles, except *"structure the UI,"* the students achieved better results when using FENIkS. When not using FENIkS, the students performed poorly in "*minimize the user memory load"* and "*speak the user language,"* and achieved the best in "*strive for consistency"* principle. For the test where the students used FENIkS, they made the most errors in "*structure the UI"* and "*minimize user's memory load" principles.* However, they made the least errors in "*allow users to use either keyboard or mouse"* and "*provide visual cues,"* which are "*to actively observe"* principles. When using FENIkS, the differences in the errors are shown in light green for fewer errors, dark green for the principles with best results, and orange for the most errors.

For Group 1, the most significant improvement by using FENIkS is achieved for "*provide visual cues"* and "*minimize user's memory load" principles*. For Group 2, the most significant improvement is achieved in "*good error messages"* and "*offer informative feedback" principles.* Generally, the most significant improvement by using FENIkS is achieved in "*provide visual cues," "offer informative feedback,"* and "*good error messages"* principles in order.

For all the "*to actively observe*" principles, each group achieved better results for the "*to actively observe*" principles when using FENIkS, and similar were observed for the entire population.

For the categories "*observed by default*" principles without teaching support, no improvement was observed when using FENIkS except for **"***offer informative feedback."*

3) Analysis of short-term persisting learning effect

Table V shows a within-group analysis, but a cross-group analysis is required to measure the persistence of the learning effects. As shown in Table VI, Group 2 performed slightly better than Group 1. However, if we test the difference using a two-sample t-test, we observe that the slight decrease in errors is insignificant (p-values > 0.1). Therefore, we have no evidence for H4.

TABLE VI
CROSS-GROUP ANALYSIS: TWO-SAMPLE T-TEST ASSUMING EQUAL VARIANCES FOR "TO ACTIVELY OBSERVE" PRINCIPLES

$\bar{X}_{error}$ without (before–G1)	$\bar{X}_{error}$ without (after–G2)	$\bar{X}_{difference}$	p-value one-tail	p-value two-tail
5.76	5.24	−0.53	0.323	0.647

4) User acceptance

We tested H5 using a questionnaire (see Section IV-B) to measure the perceived usefulness. Cronbach's alpha was 0.933, which indicates a high level of internal consistency. The results from the questionnaire on user acceptance are presented in Table VIII. The results for each group and the entire population are similar, as shown in Table VII.

V. DISCUSSION

A. Internal Validity

The experiment did not include a control group because FENIkS is used in a course where students are graded. There is a psychological risk in classroom studies: "students may worry about whether and how their participation or non-participation in the experiment will affect their grade" [68]. It is also impossible/unethical to deny half of the group access to a tool that might improve their learning. As pointed in [58], ensuring that each student receives the same value from the experiment helps satisfy an important pedagogical ethic. Therefore, in line with the ethical considerations, we conducted a quasi-experiment instead of a classic experiment in this research. The problems, as mentioned earlier, were mitigated by using a crossover design with two groups.

To avoid the observer-expectancy effect, probably present in experiments involving students [59], we ensured the following:
1. we randomly assigned each participant to groups, thus increasing the internal validity,
2. we followed the same protocol in the same way for every student in every test we performed, and
3. we used a single-blind study, where the students did not know which group they were assigned.

TABLE VII
MEAN ERROR RATES GROUP BY PRINCIPLE

Principle	Group 1 X_{error} without	Group 1 X_{error} with	Group 1 Diff.	Group 2 X_{error} without	Group 2 X_{error} with	Group 2 Diff.	Entire population X_{error} without	Entire population X_{error} with	Entire population Diff.
To actively observe									
Prevent errors	1.41	1.00	−0.41	1.53	1.41	−0.12	1.47	1.21	−0.26
Good error messages	1.41	1.24	−0.17	1.47	0.71	−0.76	1.44	0.97	−0.47
Provide visual cues	1.94	0.88	−1.06	1.00	0.88	−0.12	1.47	0.88	−0.59
Allow users to use either keyboard or mouse	1.00	0.65	−0.35	1.24	0.71	−0.53	1.12	0.68	−0.44
Observed by default - with feedback									
Structure the UI	1.82	2.00	+0.18	1.65	2.24	+.59	1.74	2.12	+0.38
Offer informative feedback	1.59	1.59	0.00	1.88	0.59	−1.29	1.74	1.09	−0.65
Strive for consistency	0.94	1.29	+0.35	1.18	0.82	−0.36	1.06	1.06	0.00
Without teaching support									
Visibility	1.59	1.18	−0.41	1.88	1.88	0.00	1.74	1.53	−0.21
Minimize user's memory load	2.65	1.29	−1.00	1.71	2.35	+0.64	2.18	1.82	−0.36
Speak the user language	1.71	2.00	+0.29	2.29	1.59	−0.70	2.00	1.79	−0.21

TABLE VIII
QUESTIONNAIRE ON PERCEIVED USEFULNESS: ITEMS AND SCORES; RANGE 1 (LOWEST) TO 7 (HIGHEST)

	Question statement	Mean			Std. dev.			Mode		
		Both	G1	G2	Both	G1	G2	Both	G1	G2
1	Using the prototype improves my understanding of UI principles.	5.79	5.81	5.76	1.15	1.24	1.06	7	7	6
2	Using the prototype makes me understand UI principles much faster.	5.70	5.50	5.88	1.09	1.06	1.08	6	6	7
3	Using the prototype improves my understanding of the relations between the conceptual model and the generated prototype components.	5.64	5.63	5.65	1.04	1.11	0.97	6	5	6
4	Using the prototype makes me understand the relations between the conceptual model and the generated prototype components much faster.	5.21	5.00	5.41	1.25	1.32	1.14	6	5	6
5	Using the prototype improves my interpretation of usability results from the generated prototype.	5.09	5.00	5.18	1.24	1.46	0.98	5	3	5
6	Using the prototype makes me interpret usability results from the generated prototype much faster.	5.18	5.19	5.18	1.31	1.47	1.15	5	4	5
7	Previewing the UI facilitated the creation of the Presentation model.	5.09	5.56	4.65	1.33	1.12	1.37	5	5	3
8	Previewing the UI showed me the effects of the chosen options on the final UI, before UI generation.	5.48	5.94	5.06	1.33	1.20	1.30	7	7	4
9	Previewing the UI helped me to decide better about design options.	5.30	5.44	5.18	1.22	1.06	1.34	5	6	5
10	Previewing the UI allowed me to visualize what the generated UI will look like and assessing the result.	5.21	5.44	5.00	1.34	1.37	1.28	6	6	4
11	Previewing the UI facilitated performing a "what-if" analysis.	5.15	5.13	5.18	1.35	1.27	1.42	6	6	4
12	If I had the choice or opportunity, I would use this tool to learn UI design principles.	5.48	5.31	5.65	1.18	1.10	1.23	6	6	7
13	If I had to vote, I would vote in favor of using prototyping in the classroom.	5.33	5.13	5.53	1.22	1.17	1.24	6	4	7
14	I am enthusiastic about using prototyping in this kind of course.	5.33	5.38	5.29	1.22	1.22	1.23	6	6	5
15	Using the prototype was a positive experience.	6.06	5.69	6.41	1.18	1.16	1.09	7	6	7

To avoid a maturation effect, the students did not receive feedback on the first test's solutions because the students' prior knowledge could impact the test results. Although students reported moderate prior knowledge and a few extensive knowledge, the initial test reveals that all students perform at the expected UI design expertise level.

Furthermore, the measurement of knowledge could also be a source of errors. In the pilot experiment, we observed that instead of an improvement, there was a decrease in some students' test scores. This decrease could be attributed to the few questions per principle, and also because the questions were true or false, students sometimes guess the answers without understanding the underlying principles. In the final experiment, we solved the problem by having at least one question per principle and asking them to write a short motivation of their answers. It enabled us to detect guessed answers for true/false questions, making the results of the final experiment more reliable.

B. *External Validity*

The validity of the results is limited to the course described in this research, and generalization beyond this course requires caution. Nevertheless, the experiment's external validity improved by making the subject population similar to the target population [58]; in this study, novice designers are the target population. The small sample size is a limitation for the pilot study. It was mitigated by replicating the experiment with a larger group of students. We performed a power analysis on our experimental design parameters, defined to detect a large effect size of at least Cohen's $d = 0.80$. During the second experiment, the sample size was adequate to identify significant improvement, with a large effect size in the performed tests in general and a statistical power of 0.87 for each group of 17 students and 0.99 for the entire group made up of 34 students.

C. *General Observations*

We evaluated the results of the question on self-reported prior knowledge using information about previous UI design knowledge collected from the participants during the experiment. Comparing this information with the curriculum used by the students, we observed that none of the students had taken a prior course on UI design, even those that reported medium to extensive knowledge. Therefore, we question the reliability of measuring previous knowledge by self-reporting and accept the evaluation performed in the experiment, which provides a more accurate measurement of the student's expertise in UI design.

Only H4 about the persistence of the learning effect could not be proved, which may be due to the short duration of the experiment. A study extending to several weeks could reveal different results with the expectation that the positive effects would persist if the treatment is applied several times consecutively.

Furthermore, the experiment results show better performance in the "*to actively observe*" and "*observed by default*" principles than for principles without teaching support. This could be explained by the feedback provided, which states why a principle is considered well applied or not helping the students understand the principles. Achieving better results in "*to actively observe*" principles than in other principles indicates that the possibility of making various choices during design, interacting with a simulation of the UI, and seeing the outcomes of the choices made is a better approach than only showing the feedback to the student.

The principle "structure the UI" had only one associated guideline in FENIkS. For the other of the principles, there were at least two associated guidelines. The limited number of associated guidelines is a possible explanation of why the students achieved the worst results for this principle.

VI. Conclusion

This paper presents and assesses FENIkS, an educational environment that combines simulation and feedback in an MDE approach to support the learning of functional UI design principles. To our knowledge, FENIkS is the first and only educational environment that supports the simulation of UIs by generating a functional prototype with feedback. FENIkS preview feature helps learners improve their design skills by quickly visualizing the outcome of various design decisions; therefore, they "learn by experience." The UI feedback features in the generated prototype help learners validate the generated UI in a fast and easy way. Furthermore, the impact of various design decisions made in the PM can be compared in the final prototype by experimenting with a concrete form of an enterprise information system. Experimental results indicate that FENIkS improved the understanding of novice designers' UI design principles, resulting in significantly improved learning outcomes. Furthermore, most students found FENIkS helpful in understanding UI design principles.

Besides contributing to the teaching of UI design, the results of this research are also applicable to instructional design. Notably, students performed better with FENIkS for the "*to actively observe*" principles than for others. It reveals that actively making choices during design and interacting with a simulation of the UI to see the effect of the choices made is a better approach than only showing explanations of well-designed UIs to the student. From an instructional design perspective, this indicates that students learn more from errors than from explanations of correct solutions. On the one hand, some research showed that worked examples could be more effective than erroneous examples for teaching problem-solving. On the other hand, simulation enables students to "experience" rather than "observe" the errors. Therefore, feedback and simulation help students because they can make changes to the options related to the principles and see the effect while interacting with the simulation of the UI and in the feedback ("*to actively observe*").

These results motivate the further development of feedback-enriched simulation tools for supporting the learning of UI and application design. Further work could cover other aspects of the design process. FENIkS could be further extended to specify a user model, which would enable the consideration of a user's skills and characteristics. The use of a user model would allow enhancing the support for learning UI design in a way that novice learners can check the outcomes of different design choices based on the user's skills and characteristics. Furthermore, we plan to generate UI for other contexts of use. Presently, FENIkS only addresses the development of enterprise information systems in one context of use. However, since this approach relies on MDE, and incorporates an AUI model, future versions of the tool can adapt the generation of the interactive software system to other contexts of use. Generating the prototype for different contexts of use will enable comparing and giving feedback based on the design output in different UIs.

Acknowledgment

The authors would like to thank Enago (www.enago.com) for the English language review.

References

[1] P. A. Akiki, A. K. Bandara, and Y. Yu, "Adaptive model-driven user interface development systems," *ACM Comput. Surv.*, vol. 47, no. 1, pp. 1–33, May 2014, doi: 10.1145/2597999.

[2] P. A. Kirschner and J. J. G. Van Merrienboer, *Ten steps to complex learning*, 3rd ed. New York, NY, USA: Routledge, 2018.

[3] H. Magrez, K. Salmi, and A. Ziyyat, "Interactive simulations for teaching and learning differential equations," in *Proc. Int. Conf. Inf. Technol Organizations Development (IT4OD)*, Feb. 2016, pp. 1-5, doi: 10.1109/IT4OD.2016.7479266.

[4] B. K. S. Khoo, "A computerized constructionist approach to simulation and modeling pedagogy," *Int. J. Manag. Inf. Syst.*, vol. 18, no. 2, pp. 87–98, Mar. 2014, doi: 10.1109/IT4OD.2016.7479266.

[5] R. Lindgren, M. Tscholl, S. Wang, and E. Johnson, "Enhancing learning and engagement through embodied interaction within a mixed reality simulation," *Comput. Educ.*, vol. 95, pp. 174–187, Apr. 2016, doi: 10.1016/j.compedu.2016.01.001.

[6] S. Kardan and C. Conati, "Providing adaptive support in an interactive simulation for learning: an experimental evaluation," in *Proc. 33rd Annu. ACM Conf. Hum. Factors Comput. Syst. (CHI '15)*, Apr. 2015, pp 3671–3680, doi: 10.1145/2702123.2702424.

[7] X. Wei, D. Weng, Y. Liu, and Y. Wang, "Teaching based on augmented reality for a technical creative design course," *Comput. Educ.*, vol. 81, pp. 221–234, Feb. 2015, doi: 10.1016/j.compedu.2014.10.017.

[8] C. De Raffaele, S. Smith and O. Gemikonakli, "The aptness of Tangible User Interfaces for explaining abstract computer network principles," in *Proc. 2016 IEEE Frontiers in Education Conference (FIE)*, Erie, PA, USA, 2016, pp. 1-8, doi: 10.1109/FIE.2016.7757573.

[9] F. Beuvens and J. Vanderdonckt, "Designing graphical user interfaces integrating gestures," in *Proc. 30th ACM Int. Conf. Design Comm. (SIGDOC '12),* Oct. 3–5, 2012, pp. 313–322.

[10] J.-S. Sottet, A. Vagner, and A. G. Frey, "Model Transformation Configuration and Variability Management for User Interface Design," in *Model-Driven Engineering and Software Development*, P. Desfray, J. Filipe, S. Hammoudi, and L. Ferreira Pires, Eds., 2015, pp. 390–404, doi: 10.1007/978-3-319-27869-8_23

[11] T. Sboui and M. B. Ayed, "Generative Software Development Techniques of User Interface: Survey and Open Issues," *Int. J. Comput. Sci. Inf. Secur.*, vol. 14, no. 7, pp. 824–842, Jul. 2016.

[12] M. L. Barrett, "A hypertext module for teaching user interface design," *SIGCSE Bull.*, vol. 25, no. 1, pp. 107–111, Mar. 1993, doi: 10.1145/169073.169359.

[13] A. Holzinger, "Usability engineering methods for software developers," *Commun. ACM*, vol. 48, no. 1, pp. 71–74, Jan. 2005, doi: 10.1145/1039539.1039541.

[14] G. Botterweck, "A model-driven approach to the engineering of multiple user interfaces," *Models in software engineering*, T. Kühne, Ed., 2007, pp. 106–115, doi: 10.1007/978-3-540-69489-2_14

[15] K.-S. Song, "Teaching software engineering through real-life projects to bridge school and industry," *SIGCSE Bull.*, vol. 28, no. 4, pp. 59–64, Dec. 1996, doi: 10.1145/242649.242667.

[16] B. K. S. Khoo, "User interface design pedagogy: a constructionist approach," in *Learning Tools and Teaching Approaches through ICT Advancements.*, L. A. Tomei, Ed., Hershey, PA, USA: IGI Global, pp. 252–261, 2013, DOI: 10.4018/978-1-4666-2017-9.ch022.

[17] J. W. van Aalst, C. van der Mast, and T. T. Carey, "An Interactive Multimedia Tutorial For User Interface Design," *Comput. Educ.*, vol. 25, no. 4, pp. 227–233, Dec. 1995, doi: 10.1016/0360-1315(95)00076-3.

[18] J. J. G. Van Merriënboer, R. E. Clark, and M. B. M. De Croock, "Blueprints for complex learning: the 4C/ID-model," *Educ. Technol. Res. Dev.*, vol. 50, no. 2, pp. 39–61, Jun. 2002, doi: 10.1007/BF02504993.

[19] N. Fenton, "Viewpoint article: conducting and presenting empirical software engineering," *Empir. Softw. Eng.*, vol. 6, pp. 195–200, Sep. 2001, doi: 10.1023/A:1011449731678.

[20] J. Hattie and H. Timperley, “The power of feedback,” *Rev. Educ. Res.*, vol. 77, no. 1, pp. 81–112, Mar. 2007, doi: 10.3102/003465430298487.

[21] P. Black and D. Wiliam, "Inside the black box: raising standards through classroom assessment," *Phi Delta Kappan*, vol. 92, no. 1, pp. 81–90, Sep. 2010, doi: 10.1177/003172171009200119

[22] A. G. Sutcliffe, S. Kurniawan, and J.-E. Shin, “A method and advisor tool for multimedia user interface design,” *Int. J. Hum. Comput. Stud.*, vol. 64, no. 4, pp. 375–392, Apr. 2006, doi: 10.1016/j.ijhcs.2005.08.016.

[23] F. B. V. Benitti and L. Sommariva, “Evaluation of a game used to teach usability to undergraduate students in computer science,” *J. Usability Stud.*, vol. 11, no. 1, pp. 21–39, Nov. 2015.

[24] A. Lisowska Masson, D. Lalanne, and T. Amstutz, “A usability refactoring process for large-scale open source projects: the ILIAS case study,” in *Proc.2017 CHI Conf. Extended Abstr. Human Factors Comp. Syst.*, May 2017, pp 1135–1143, doi: 10.1145/3027063.3053345

[25] W. O. Galitz, *The essential guide to user interface design: an introduction to GUI design principles and techniques*. New York, NY, USA: John Wiley & Sons, 2007.

[26] B. Emond and R. L. West, “Using cognitive modelling simulations for user interface design decisions,” in *Proc. 17th Int. Conf. Innovations in Appl. Artif. Intell.*, Ottawa, Canada, 2004, pp. 305–314.

[27] P. K. A. Wollner, P. M. Langdon, and P. J. Clarkson, “Integrating a cognitive modelling framework into the design process of touchscreen user interfaces,” in *Design, User Experience, and Usability: Users and Interactions*, A. Marcus, Ed., Heidelberg, Germany: Springer, 2015, pp. 473–484.

[28] T. Clerckx, K. Luyten, and K. Coninx, “DynaMo-AID: A design process and a runtime architecture for dynamic model-based user interface development,” in *Engineering Human Computer Interaction and Interactive Systems*, R. Bastide, P. Palanque, and J. Roth, Eds., Heidelberg, Germany: Springer, 2005, pp. 77–95.

[29] H. Trætteberg, “A hybrid tool for user interface modeling and prototyping,” in *Computer-Aided Design of User Interfaces*, V. G. Calvary, C. Pribeanu, G. Santucci, and J. Vanderdonckt,Eds., Heidelberg, Germany: Springer, 2007, pp. 215–230.

[30] A. M. R. da Cruz and J.P. Faria, “Automatic generation of user interface models and prototypes from domain and use case models,” in *Proc. 6th Int. Conf. Qual. Inf. Commun. Technol. (QUATIC 2007)*, Lisbon, 2007, pp. 208-212, doi: 10.1109/QUATIC.2007.19

[31] F. Montero and V. López Jaquero, “GUILayout++: supporting prototype creation and quality evaluation for abstract user interface generation,” in Proc. 1st. USer Interface eXtensible Markup Lang. Workshop (UsiXML-EICS 2010). Berlin, Germany, June 20, 2010, pp 39-44

[32] D. Raneburger, R. Popp, and J. Vanderdonckt, “An automated layout approach for model-driven wimp-ui generation,” in *Proc. 4th ACM SIGCHI Symp.Eng. Interact. Comput. Syst. (EICS '12)*, Copenhagen, Denmark, 2012, pp. 91–100.

[33] A. I. Molina, W. J. Giraldo, J. Gallardo, M. A. Redondo, M. Ortega, and G. García, “CIAT-GUI: a mde-compliant environment for developing graphical user interfaces of information systems,” *Adv. Eng. Softw.*, vol. 52, pp. 10–29, Oct. 2012, doi: 10.1016/j.advengsoft.2012.06.002.

[34] B. Khoo and J. Preece, “An interactive case scenario for teaching user interface design,” in *AMCIS 1999 Proc.,* 1999, [Online]. Available: https://aisel.aisnet.org/amcis1999/321.

[35] D. Boud and E. Molloy, “Rethinking models of feedback for learning: the challenge of design,” *Assess. Eval. High Educ.*, vol. 38, no. 6, pp. 698–712, Mar. 2013, doi: 10.1080/02602938.2012.691462.

[36] J. Albors-Garrigos and J. C. R. Carrasco, “New learning paradigms: open course versus traditional strategies. the current paradox of learning and developing creative ideas,” in *Social media tools and platforms in learning environments*, B. White, I. King, and P. Tsang, Eds., Berlin, Germany: Springer-Verlag, 2011, pp. 53–79.

[37] J. Barjis, A. Gupta, R. Sharda, T. Bouzdine-Chameeva, P. D. D. Lee, and A. Verbraeck, “(GbL #3) Innovative teaching using simulation and virtual environments,” *Interdiscip. J. Inf. Knowl. Manag.*, vol. 7, pp. 237–255, 2012, doi: 10.28945/1750.

[38] A. Kluge, “Experiential learning methods, simulation complexity and their effects on different target groups,” *J. Educ. Comput. Res.*, vol. 36, no. 3, pp. 323–349, Apr. 2007, doi: 10.2190/B48U-7186-2786-5429.

[39] D. A. Damassa and T. D. Sitko, “Simulation technologies in higher education: uses, trends, and implications,” *ECAR Res. Bull.*, vol. 3, Feb. 2010.

[40] K. Koukouletsos, B. Khazaei, A. Dearden, and M. Ozcan, “Teaching usability principles with patterns and guidelines,” in *Creativity and HCI: From Experience to Design in Education*, P. Kotzé, W. Wong, J. Jorge, A. Dix, and P. A. Silva, Eds., Boston, USA: Springer, 2009, pp. 159–174, doi: 10.1007/978-0-387-89022-7_11.

[41] J. Ruiz, E. Serral, and M. Snoeck, “A fully implemented didactic tool for the teaching of interactive software systems,” in *Proc. 6th Int. Conf. Model-Driven Eng. Soft. Dev.*, S. Hammoudi, L. Ferreira Pires, and B. Seli, Eds., Jan. 2018, pp. 95–105.

[42] J. Ruiz, E. Serral, and M. Snoeck, “UI-GEAR: User Interface Generation prEview capable to Adapt in Real-time,” in *Proc. 5th Int. Conf. Model-Driven Eng. Soft. Dev.,* L. Ferreira Pires, S. Hammoudi, and B. Seli, Setubal, Eds., Feb. 2017, pp. 277–284.

[43] J. Ruiz, G. Sedrakyan, and M. Snoeck, “Generating user interface from conceptual, presentation and user models with jmermaid in a learning approach,” in *Proc. 16th Int. Conf. Human Comput. Interaction (Interacción '15)*, New York, USA, 2015, pp. 1–8. doi: 10.1145/2829875.2829893

[44] T. Mandel, *The elements of user interface design*, vol. 20. New York, NY, USA: Wiley, 1997.

[45] J. Hattie and S. Clarke, *Visible learning: feedback*. New York, NY, USA: Routledge, 2018.

[46] P. L. Smith and T. J. Ragan, *Instructional design*, 3rd ed. Hoboken, NJ, USA: Wiley, 2004.

[47] G. Meixner, F. Paternò, and J. Vanderdonckt, “Past, present, and future of model-based user interface development,” *i-com*, vol. 10, no. 3, pp. 2–11, Nov. 2011, doi: 10.1524/icom.2011.0026.

[48] K. D. Nguyen and M. A. Rahman, “Identifying interface design patterns by studying intrinsic designs,” in *Proc. 3th Int. Conf. Comput. Science., Comput. Eng. Educ. Technol. (CSCEET2016)*, Lodz, Poland, 2016, pp. 13–24.

[49] J. Dehinbo, “Establishing and applying criteria for evaluating the ease of use of dynamic platforms for teaching web application development,” *Inf. Syst. Educ. J.*, vol. 9, no. 5, pp. 86–96, Oct. 2011.

[50] G. Sedrakyan, M. Snoeck, and S. Poelmans, “Assessing the effectiveness of feedback enabled simulation in teaching conceptual modeling,” *Comput. Educ.*, vol. 78, pp. 367–382, Sep. 2014, doi: 10.1016/j.compedu.2014.06.014.

[51] G. Sedrakyan and M. Snoeck, “A PIM-to-code requirements engineering framework,” in *Proc. 1st Int. Conf. Model-Driven Eng. Soft. Dev.,* Barcelona, Spain, Feb. 2013, pp. 277–284.

[52] M. Snoeck, *Enterprise Information Systems Engineering: The MERODE Approach*, Heidelberg, Germany: Springer, 2014.

[53] G. Sedrakyan, M. Snoeck, and J. De Weerdt, “Process mining analysis of conceptual modeling behavior of novices-empirical study using JMermaid modeling and experimental logging environment,” *Comput. Hum. Behav.*, vol. 41, pp. 486–503, Dec. 2014, doi: 10.1016/j.chb.2014.09.054.

[54] G. Sedrakyan and M. Snoeck, “Feedback-enabled MDA-prototyping effects on modeling knowledge,” in *Enterprise, Business-Process and Information Systems Modeling*, S. Nurcan et al., Eds., 2013, pp. 411–425, doi: 10.1007/978-3-642-38484-4_29

[55] E. Serral, J. Ruiz, J. Elen, and M. Snoeck, “Conceptualizing the domain of automated feedback for learners,” in *Proc. 22nd Ibero-American Conf. Soft. Eng.*, La Habana, Cuba, 2019, pp. 223–236.

[56] B. S. Bloom and Committee of College and University Examiners, *Taxonomy of educational objectives,* Vol. 2, London, UK: Longmans, Green & Co Ltd, 1964.

[57] S. Poelmans and P. Wessa, “A constructivist approach in an e-learning environment for statistics: a students’ evaluation,” *Interact. Learn. Environ.*, vol. 23, no. 3, pp. 385–401, May 2015, doi: 10.1080/10494820.2013.766890.

[58] J. C. Carver, L. Jaccheri, S. Morasca, and F. Shull, “A checklist for integrating student empirical studies with research and teaching goals,” *Empir*, *Softw. Eng.*, vol. 15, pp. 35–59, 2010, doi: 10.1007/s10664-009-9109-9.

[59] A. J. Ko, T. D. LaToza, and M. M. Burnett, “A practical guide to controlled experiments of software engineering tools with human participants,” *Empir*, *Softw. Eng.*, vol. 20, pp. 110–141, 2015, doi: 10.1007/s10664-013-9279-3.

[60] J. R. Lewis “IBM computer usability satisfaction questionnaires: psychometric evaluation and instructions for use,” *Int. J. Hum.-Comput. Int.*, vol. 7, no. 1, pp. 57–78, Jan. 1995, doi: 10.1080/10447319509526110.

[61] C. M. Stoney and L. L. Johnson, “Design of clinical trials and studies,” *Principles and Practice of Clinical Research*, 4th ed., J. Gallin, F.

Ognibene, L. L. Jackson, Eds., Cambridge, UK: Academic Press, 2017, pp. 249–268.

[62] J. C. Chen, D. C. Whittinghill, and J.A. Kadlowec, "Using Rapid Feedback To Enhance Student Learning And Satisfaction," in *Proc. 36th Annu. Conf. Frontiers in Educ.*, San Diego, CA, 2006, pp. 13–18, doi: 10.1109/FIE.2006.322306.

[63] M. Bose and A. Dey, "Developments in crossover designs," Indian Statistical Institute, New Delhi, India. Accessed: Sep. 2, 2020. [Online]. Available: https://www.semanticscholar.org/paper/Developments-in-Crossover-Designs-Bose-Dey/0a85e92e6464b58a8a17eb1f2956edf5e9702066. 2013.

[64] G. Sedrakyan, S. Poelmans, and M. Snoeck, "Assessing the influence of feedback-inclusive rapid prototyping on understanding the semantics of parallel uml statecharts by novice modellers," *Inf. Softw. Technol.*, vol. 82, pp. 159–172, Feb. 2017, doi: 10.1016/j.infsof.2016.11.001.

[65] R. Shier, "Statistics: 1.1 Paired T-tests," *Math. Learn. Support Cent.*, [Online]. Available: http://www.statstutor.ac.uk/resources/uploaded/paired-t-test.pdf.

[66] N. M. Razali and Y. B. Wah, "Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests," *J. Stat. Model Anal.*, vol. 2, no. 1, pp. 21–33, Jan. 2011.

[67] H. C. Thode, *Testing for normality*. New York, NY, USA: Taylor & Francis, 2002.

[68] J. E. Sieber, "Protecting research subjects, employees and researchers: implications for software engineering," *Empir. Softw. Eng.*, vol. 6, pp. 329–341, Dec. 2001, doi: 10.1023/A:1011978700481.

Jenny Ruiz was born in Holguin, Cuba in 1981. She received the B.S. and M.S. degrees in informatics engineering and applied mathematics and informatics for management from the University of Holguin, Cuba, in 2004 and 2007 and the Ph.D. degree in business economics from the KU Leuven, Belgium, in 2018.

From 2010 to 2018, she was Assistant Professor with the Informatics Engineering Department, of the Faculty of Informatics and Mathematics, University of Holguin. Since 2018 she has been Full Professor with the same department. Her research interests include User Interface design, model-driven engineering and software engineering.

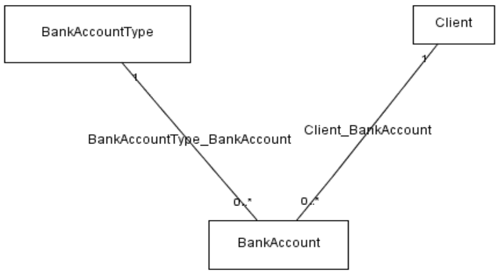
Estefanía Serral obtained her PhD in computer science in 2011 from the Technical University of Valencia, Spain. She is currently Assistant Professor at KU Leuven (Belgium). Previously, she was Assistant Professor at TU/e, The Netherlands. From 2012 to 2014, she led the Semantic Knowledge Representation and Integration research group at the CDL-Lab at the TU Vienna (Austria). Until 2012, she worked in the ProS Research Center at the Technical University of Valencia (Spain), where she designed a novel method for developing ubiquitous systems using model-driven development and semantic technologies. Prof. Serral has many publications in high-ranking conferences and journals, such as CAiSE, ER, UIC, PMC, ESWA, SOSYM, MTAP, etc. She is currently doing research in topics such as Internet of Things, ubiquitous business processes, and context-adaptive systems.

Monique Snoeck (M'87) received the PhD. degree in computer science in 1995 from KU Leuven, Belgium. She is currently Full Professor at KU Leuven, Head of the Research Center on Information Systems Engineering, and Visiting Professor at UNamur. She has published over 200 papers in highly ranked journals and conferences. Her interests include conceptual modeling, enterprise architecture, model-driven engineering and technology enhanced learning.

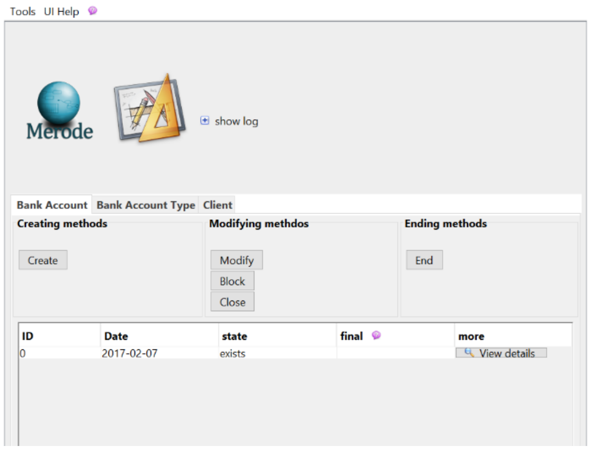
APPENDIX A EXERCISES

A. *Exercise A*

This model presents a system to manage the information about people and their bank accounts. With that system, you can create a person, a bank account type, a bank account associated to a person, modify the bank account, block it or close it. You can also end (change to a final state) all of the objects.

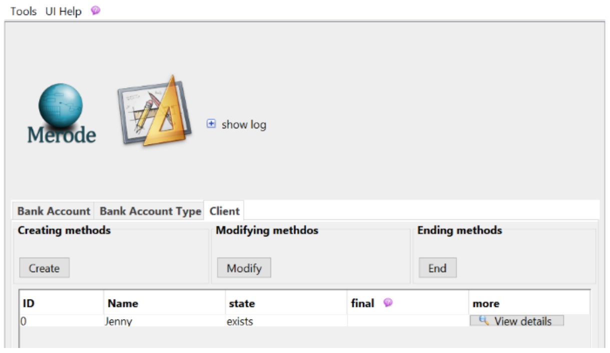


The following screenshots correspond to the generated system.
1. Main window of the application. You are in Bank Account tab



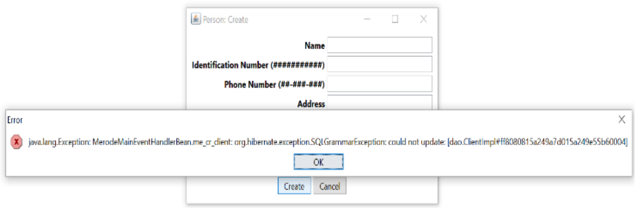
ID	Date	state	final	more
0	2017-02-07	exists		View details

2. Main window of the application. You are in the Client tab.



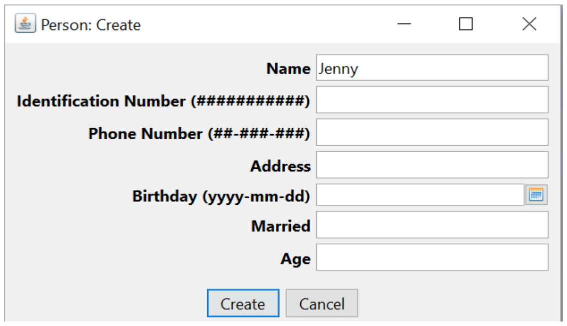
ID	Name	state	final	more
0	Jenny	exists		View details

3. Window to create a person. You clicked on Create button.


Person: Create
Name
Identification Number (###########)
Phone Number (##-###-###)
Address
Error
OK
Create
Cancel

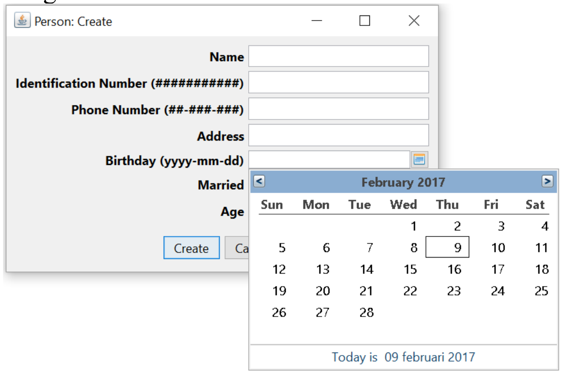
java.lang.Exception: MerodeMainEventHandlerBean.me_cr_client: org.hibernate.exception.SQLGrammarException: could not update: [dao.ClientImpl#ff8080815a249a7d015a249e55b60004]

4. Window to create a person. You entered the name of a person.



Name	Jenny
Identification Number (###########)	
Phone Number (##-###-###)	
Address	
Birthday (yyyy-mm-dd)	
Married	
Age	

5. Window to create a person. You clicked on the birthday widget.

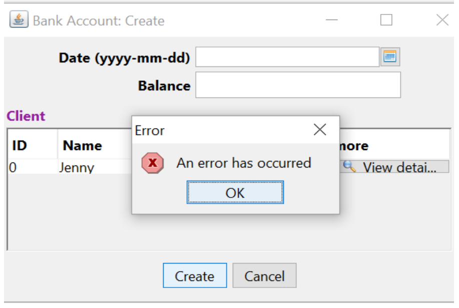


February 2017

Sun	Mon	Tue	Wed	Thu	Fri	Sat
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28				

Today is 09 februari 2017

6. Window to create a Bank Account. You clicked on Create button.



ID	Name	more
0	Jenny	View detai...

An error has occurred

Answer the following questions:
1. Select the correct answer. How can you make the application easier and quicker to use for experienced users? Why? (one is correct).
   - By providing shortcuts for the frequent actions a user can make.
   - By minimizing the visual cues for known options.

2. In the system, the user needs to enter his/her identification number (see screenshot 4). Is the way it is asked to enter the identification number correct? Why?
3. Select the correct answer. The principle "structure the user interface" is applied in screenshot 5 by (one is correct):
   - Classifying/Grouping the methods the user can perform for the Bank Account.
   - Hiding some attributes of the client and allowing seeing them with the View details button.
   - None of the previous.

Explain your answer.

4. Which of the following are examples of feedback:
- The sound of the keyboard when typing.
- The name of the link you can click.
- The progress bar.
- The position of mouse.

5. In screenshot 5, the identification number of the client is asked for and a few fields lower, the date of birth is asked for. This is an example of the violation of:
- Offer informative feedback.
- Prevent errors.
- Strive for consistency.
- Minimize user´s memory load.

Explain your answer.

6. Select the correct answer. Having a title for each window is an example of the principle (one is correct):
- Organization.
- Structure the user interface.
- Visibility.
- Provide visual cues.

7. In a system, the user needs to enter his/her matric number. Select the best way to show that to the user:
- Enter your Matric Number (XXX12345): ________
- Enter your Matric Number: ______

8. Which of the following does not reduce the user's memory load? (one is correct)__
- Define intuitive shortcuts.
- Disclose information in a progressive fashion.
- Provide an online tutorial.
- Stablishing meaningful defaults.

9. A user specified a telephone number incorrectly on a data entry screen. Select the correct error message:
- Invalid number.
- Sorry, you entered too few digits. You need to enter a 10 digit number. Please try again.

10. Select the correct answer. When you write your name it is shown in the text box name, as in the screenshot 4. This is an example of the application of which of the following principles (one is correct):
- Organization.
- Feedback.
- Visibility.
- Provide visual cues.

Explain your answer.

11. Select the correct answer. "Making visible all and only the information the user needs to complete a task at hand" is an example of the following principle (one is correct):
- Offer informative feedback.
- Prevent errors.
- Visibility.
- Minimize user´s memory load.

12. One of the data items to be entered in screenshot 5 is the birthday of the client. Is it correct the way it is asked? Explain your answer.

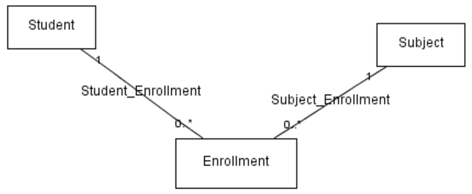
13. Indicate whether the following statements are true or false:

Nr	Statement	True Or False?
1	For the users it is important that the user interface provides them with handy shortcuts for important tasks.	
2	When the components of an interface are generated as in the screenshots, it is not possible to prevent errors.	
3	When several options are presented, their organization must be logical.	
4	When the user interface has the widgets grouped in sections the consistency is lost.	
5	The "Visibility" principle is applied in the presented system because the system uses the same font in all the interfaces.	
6	Allowing users entering the words True or False for the Boolean value of the field "Married" helps users to prevent errors.	
7	Prompts for data or command entry should be displayed in a standard location.	
8	The error message shown in screenshot 4 is a good error message.	
9	When a list of options is shown, the options should be always organized alphabetically.	
10	In order to be consistent, the user interface should only use the keyboard or the mouse.	
11	A general principle of user interface design is to use as much as possible detailed explanation when an error occurs.	
12	"Visibility" principle is applied in the presented system because the system provides a distinction between the zones to input data and the zones where data is shown.	
13	Showing one part of the information of the user interface in one language and the other in other language helps different users to better understand the application.	
14	You are changing the way user accesses the methods: You should use menus for the creating methods and buttons for the modifying methods.	
15	The "Visibility" principle is applied in the presented system because the system shows information in clear tables.	
16	The error message shown in screenshot 3 is a good error message.	

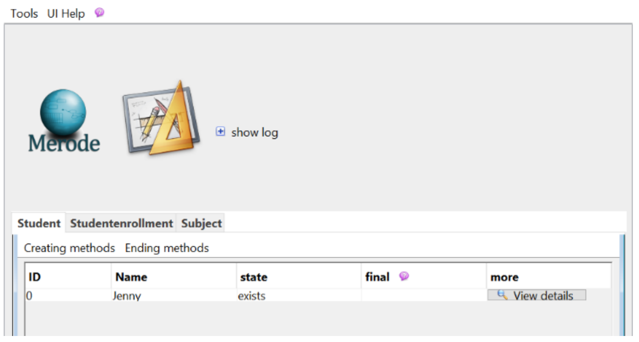
For the statements 13, 14, 15 and 16 explain your answers.

B. *Exercise B*

This model presents a system to manage the information of students and the subjects they are enrolled for. With this system you can create a student, a subject, enroll a student to a subject, modify the enrollment, postpone it and suspend it. You can also end (change to a final state) all of the objects.

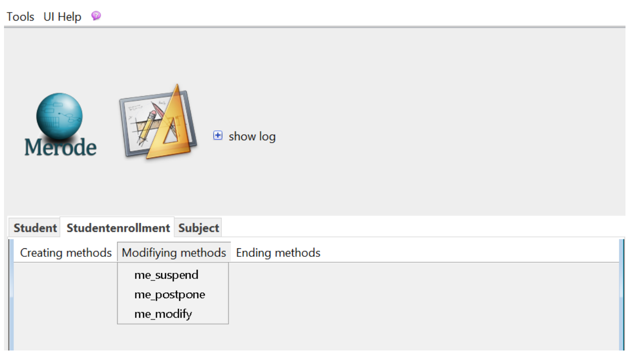


The following screenshots correspond to the generated system.
1. Main window of the application. You are in the Student tab

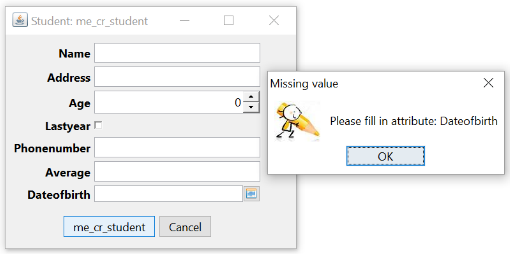

Tools UI Help
Merode
show log
Student Studentenrollment Subject
Creating methods Ending methods

ID	Name	state	final	more
0	Jenny	exists		View details

2. Main window of the application. You are in StudentEnrollment tab



3. Window to create a student. You clicked on me_cr_student item of the Creating methods menu.

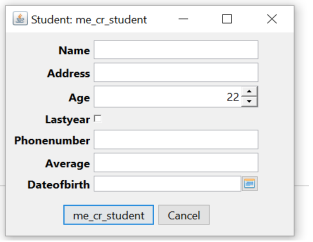


Age 0

Missing value

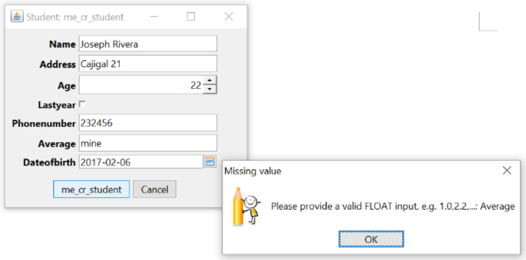
Please fill in attribute: Dateofbirth

4. Window to create a student. You clicked on the age widget.



Age 22

5. Window to create a Student. You clicked on me_cr_student button.



Name Joseph Rivera
Address Cajigal 21
Age 22
Lastyear
Phonenumber 232456
Average mine
Dateofbirth 2017-02-06

Missing value

Please provide a valid FLOAT input, e.g. 1.0,2.2,...: Average

Answer the following questions:
1. In the system the user needs to enter his/her phone number. Is the way it is asked in screenshot 5 correct? Why?
2. Select the correct answer. The generated system presents methods classified in creating, modifying and ending methods. This is an example of the application of the principle (one is correct):
- Organization.
- Structure the user interface.
- Consistency.
- Visibility.

Explain your answer.
3. Which of the following are examples of feedback:
- The list of pages you can visit.
- Changing the color of already visited links.
- The confirmation message when you enter an information item.
- The window's title.

For the second item explain your answer.
4. In screenshot 5 the systems asks for entering the date of birth and also the age of the student. This is an example of the violation of:
- Offer informative feedback.
- Prevent errors.
- Strive for consistency.
- Minimize user´s memory load.

Explain your answer.
5. Select all the correct answers. The design principle "Strive for consistency" implies that:
- Each application should have its own distinctive look and feel.
- Input mechanisms remain the same throughout the application.
- Navigational methods are context sensitive.
- Visual information is organized according to a design standard.

6. Select the correct answer. The principle "structure the user interface" can be applied by (one is correct):
- Showing appropriated images corresponding to the text that is shown in the interface.
- Organizing items in hierarchical lists.

7. When creating a new student, select the attributes that need visual cues:
- Date of birth.
- Age.
- Phone number.
- Address.

Provide an example.
8. Select the correct answer. When you enter a new student the name is shown in a row of a table, as in the screenshot 1. This is an example of the application of the principle (one is correct):
- Organization.
- Feedback.
- Visibility.
- Provide visual cues.

Explain your answer.
9. One of the data items to be entered is the age of the student in screenshot 5. Is the way it is asked correct? Explain your answer.
10. Actions that are accessed in similar way, related controls that are grouped together, and messages that uses a uniform structure are examples of which UI design principle?
- Permit easy reversal of actions.
- Design dialogs to yield closure.
- Offer informative feedback.

11. You are designing a research submarine for underwater science and exploration. You are told that your users will all have PhDs in marine biology. Is it appropriate to use terminology/metaphors from this field? Explain your answer.
12. Indicate whether the following statements are true or false:

Nr	Statement	True Or False?
1	The system should allow experienced users by passing a series of menu selections and making an equivalent command entry or using keyboard shortcuts.	

2	The error message shown in screenshot 3 is a good error message.
3	It is better to give the users the largest number of choices.
4	When the components of an interface are generated as in the screenshots, it is easier to prevent errors.
5	Allowing the users using either the mouse or the keyboard in the system is an example of inconsistency.
6	Providing clear visual distinction of data fields and their labels is a violation of consistency.
7	The error message shown in screenshot 5 is a good error message.
8	When showing abbreviations to the users it is better to make them as short as possible.
9	Showing appropriated images associated to the text in the interface helps the visibility of the system.
10	When introducing a name for a file, the field should be prepopulated with the old name.
11	Using technical vocabulary in a system makes it difficult to understand the system.
12	Novice users should never be allowed using the keyboard for error prevention reasons.
13	Command zone and message zone should be represented in the same way.
14	Allowing users to select the value for the field "Last year" helps users to prevent errors.
15	Showing the menus and window titles with the name of the methods in the system (rather than an alias) help users to understand how the system is built. See screenshots 2 and 3.
16	The presented screenshots show a system that is not consistent.

For the statements 12, 13, 14, 15 and 16 explain your answers.

APPENDIX B QUESTIONNAIRE TO MEASURE PERCEIVED USEFULNESS

1= strongly disagree; 2 = disagree; 3 = disagree somewhat; 4= neutral; 5 = agree somewhat; 6 = agree; 7 = strongly agree

Question	Evaluation
Using the prototype improves my understanding of User Interface principles.	O1 O2 O3 O4 O5 O6 O7
Using the prototype makes me understand User Interface principles much faster.	O1 O2 O3 O4 O5 O6 O7
Using the prototype improves my understanding of the relations between the conceptual model and the generated prototype components.	O1 O2 O3 O4 O5 O6 O7
Using the prototype makes me understand the relations between the conceptual model and the generated prototype components much faster.	O1 O2 O3 O4 O5 O6 O7
Using the prototype improves my interpretation of usability results from the generated prototype.	O1 O2 O3 O4 O5 O6 O7
Using the prototype makes me interpret usability results from the generated prototype much faster.	O1 O2 O3 O4 O5 O6 O7
Previewing the UI facilitated the creation of the Presentation model	O1 O2 O3 O4 O5 O6 O7
Previewing the UI showed me the effects of the chosen options on the final UI, before UI generation	O1 O2 O3 O4 O5 O6 O7
Previewing the UI helped me to decide better about design options	O1 O2 O3 O4 O5 O6 O7
Previewing the UI allowed me to visualize how the generated UI will look like and assessing the result	O1 O2 O3 O4 O5 O6 O7
Previewing the UI facilitated performing a "what-if" analysis	O1 O2 O3 O4 O5 O6 O7
If had the choice, or opportunity I would use this tool to learn User Interface design principles.	O1 O2 O3 O4 O5 O6 O7
If I had to vote, I would vote in favor of using prototyping in the classroom	O1 O2 O3 O4 O5 O6 O7
I am enthusiastic about using the prototyping in this kind of courses	O1 O2 O3 O4 O5 O6 O7
Using the prototype was a positive experience	O1 O2 O3 O4 O5 O6 O7

APPENDIX C CONTEXT INFORMATION

Previous knowledge (in terms of having User Interface design and/or programming course(s) before)

Previous knowledge on User Interface design in previous degree	O1 O2 O3 O 4
Previous knowledge on programming in previous degree	O1 O2 O3 O 4
Previous knowledge on software testing in previous degree	O1 O2 O3 O 4

1: no knowledge/experience at all;
2: little knowledge (a few hours course);
3: moderate knowledge (intermediate level course);
4: extensive knowledge (advanced course(s))

Years of programming experience (if applicable):______

I could use a new software application well …

… even if I had never used an application like it before.	O1 O2 O3 O4 O5 O6
… if I had just the built-in-help facility or manual for assistance.	O1 O2 O3 O4 O5 O6
… if I had first seen someone else using it before trying it myself.	O1 O2 O3 O4 O5 O6
… using only the internet for assistance.	O1 O2 O3 O4 O5 O6

1: not at all confident	4: rather yes
2: probably not	5: likely yes
3: rather not	6: totally confident: yes

On Average, I use computers (laptop, desktop, tablet) per day:
□ less than one hour
□ one to two hours
□ three to five hours
□ to eight hours
□ eight or more hours

APPENDIX D DISTRIBUTION OF THE DIFFERENCES OF ERRORS.

TABLE IX
DISTRIBUTION OF ERRORS BEFORE AND AFTER TREATMENT

	Group 1			Group 2		
Student	$\bar{X}_{error\ without}$	$\bar{X}_{error\ with}$	$\bar{X}_{difference}$	$\bar{X}_{error\ without}$	$\bar{X}_{error\ with}$	$\bar{X}_{difference}$
1	15	8	-7	12	12	0
2	20	18	-2	16	6	-10
3	9	4	-5	11	10	-1
4	31	28	-3	11	8	-3
5	11	9	-2	11	5	-6
6	23	20	-3	16	12	-4
7	17	13	-4	9	10	1
8	27	20	-7	10	9	-1
9	24	20	-4	19	20	1
10	15	16	1	20	19	-1
11	21	19	-2	24	24	0
12	9	6	-3	16	21	5
13	8	8	0	13	9	-4
14	4	6	2	18	14	-4
15	8	7	-1	22	14	-8
16	19	12	-7	23	20	-3
17	12	9	-3	18	11	-7