Journal of Computer and Education Research
March 2024 Volume 12 Issue 23
http://dergipark.org.tr/jcer
(ISSN:2148-2896)

Research Article

Integrating Usability into Software Engineering Course Projects

Nihal MENZİ ÇETİN [1]*

[1] *Adana Alparslan Türkeş Science and Technology University, Adana, Turkey* ncetin@atu.edu.tr

* *Corresponding Author:* ncetin@atu.edu.tr

Article Info

Received: *09 January 2024*
Accepted: *17 March 2024*

Keywords: Usability evaluation, use case modeling, software engineering education, prototype development

10.18009/jcer.1417245

Publication Language: English

Abstract

This study presents a case study of the integration of usability evaluation into the development of task based information system prototypes in undergraduate software engineering course projects within a problem-based learning (PBL) approach. As part of the course, usability evaluation was integrated into the software development process, and the usability evaluation performance (UEP) of the projects was assessed in terms of the following criteria: effectiveness, efficiency and problem validity. It also analyzed the problem solution rate of the projects and their correlation between UEP criteria. Additionally, the study in-depth analyzed the students' final conclusions regarding the design process. The relationship between UEP criteria and the problem solving rate was examined using Spearman correlation analysis. In addition, final conclusions and problem solving behaviors were analyzed through document and thematic analysis techniques. The results supported that there is a significant correlation between usability evaluation criteria and problem solving rate. The main themes that emerged regarding usability problem solving behaviors were navigation design, error handling, database connection, algorithm design, search matching, system-user communication design and others. Final conclusions showed that project teams gained valuable insights into user-centered design, solved critical usability problems and improved their prototype design. It can be concluded that the integration of usability into software engineering education using the above approach contributes to students' understanding of user-centered interaction design.

OPEN ACCESS CrossMark

To cite this article: Menzi-Çetin, N. (2024). Integrating usability into software engineering course projects. *Journal of Computer and Education Research, 12* (23), 209-240. https://doi.org/10.18009/jcer.1417245

Introduction

Usability standards provide requirements and recommendations for human-centered design principles and activities throughout the software development life cycle (SDLC) (Fischer, 2012). In the design of user task automation or task based information system, which is an information management system that allows the user to manage information, the main issue of human-computer interaction is the adaptation of a computer system to the needs of users (Puerta, 1996). The concern for usable and interactive system design places emphasis on usability for interaction design and graphical user interface (GUI) design. User-centered design involves the user in the design; this type of approach also reduces user effort during interaction (Velmourougan et al., 2014). Software organizations incorporate usability activities into their software development processes and reap the benefits of usability in terms of product quality, user satisfaction and competitiveness (Ardito, 2011). As an organic part of a software development process with a user interface, usability commits to the quality of the software product and influences the activation of software features (Madan & Dubey, 2012). The aforementioned concerns point to the need to integrate usability into software design.

In user-centered design, designers involve users in the design process through a variety of research and design techniques to create highly usable and accessible products for them (URL). Involving the user in the lifecycle process at a later stage adds cost and time (Liaghati, et al., 2020). Studies have shown that considering usability at the prototype design stage means fewer usability problems in the final design (Kuhnel, et al., 2018). The software engineering course project in this study aims to integrate usability into the prototype development process to help students adopt a user-centered approach at an inexperienced stage of software development, and also to learn usability evaluation and conduct high-quality evaluations.

If usability evaluation is not well integrated into the design or does not cover all aspects of usability, the work done will fail (Gulati & Dubey, 2012). We can say that quality ensures that practical results are obtained from usability evaluation. Bolchini and Garzotto (2007) highlighted the methodological quality of usability evaluation in interactive systems in terms of more measurable attributes such as performance, efficiency, cost-effectiveness and learnability. Among these factors, performance is described as an indicator of how well the usability evaluation method detects problems. Hartson et al. (2003) proposed 3 criteria for performance of usability evaluation method (UEM): thoroughness, validity and reliability. Of these, validity was preferred in terms of its suitability for studies of user information systems. In the study, usability assessments focused on three basic skills for students to acquire and evaluate as course outcomes; effectiveness, efficiency and problem validity.

Prototype development is a learning process that leads to the design of final products by utilizing the learning from this stage. Hartson et al. (2003) state that "formative evaluation focuses on usability problems that need to be solved during the prototype design stage before a final design can be accepted for release". At this point it is important to conduct an effective usability evaluation in order to guide the designer.

Problem-based learning (PBL) is a teaching strategy that allows students to apply knowledge, open-ended, contextualized and real-world situations are presented to them (Brodie, et al., 2008). In this method, students working in collaborative groups learn by solving complex, real-life problems under the guidance of faculty (Allen, et al., 2011). Implementing the PBL in the software engineering courses, Mitchell and Delaney (2004) reported the difficulty of assessment in the PBL group projects, and to overcome this problem, these four elements need to be assessed: group performance, individual contribution to the group, project outcomes and course success. This study attempted to provide a general framework by evaluating the learning outcomes of the PBL approach used in this study in a multidimensional way, as suggested in the literature.

Software Engineering Education

Ghezzi and Mandrioli (2005) explained some challenges in software engineering education; the integration of real projects in class is a critical issue, it is difficult to simulate real situations in class. It was therefore considered crucial to find innovative ways of integrating software projects into the curriculum. Project-based learning and problem-based learning approaches could meet this need by allowing students to solve real-life problems. The combination of these two approaches, the interaction of team members in providing the required outputs of the project, positively affects learning in software engineering courses (Brodie, et al., 2008). Zainol and Almukadi (2020) also verified that PBL and real-life contextualization promotes student learning and their perceptions about the course. In the context of this study, the main objectives of the course are to carry out collaborative project work to design a task based information system prototype that will meet daily needs, as well as to develop a user-centered design approach in the student's mindset. In order to achieve the second objective, a PBL approach based on completing the design by solving usability problems was adopted.

Usability in the Software Engineering Education

Software Engineering (SE) education is concerned with software development processes and models, and aims to provide undergraduate students with the theoretical and practical aspects of software development. The discipline of Human Computer Interaction (HCI), which focuses on identifying post-design interaction problems and improving the user experience, is closely related to software engineering. HCI encompasses activities throughout the software life cycle in SE processes concerned with the design of an interactive system (Dix, et al., 2004). Introducing this discipline to students during SE education will help them to adopt a user-centered approach to software design. Therefore, integrating usability studies into SE education will increase students' acquisition. By systematically reviewing the literature, Cico et al. (2021) reported that usability is one of the most prominent SE education trends and practices in studies. Boehm (2005) also put users and end value first among the top nine trends in SE practices by 2025. Although there are several SDLC models that address usability at different levels, it is very difficult to integrate usability into software engineering practices (Gupta, et al., 2017).

Usability Evaluation

Usability evaluation takes place in roughly two basic ways: evaluation through expert analysis and user participation (Dix, et al., 2004). There are several types of usability evaluation of information systems; user testing is the most widely used technique in the literature (Bernhaupt, et al., 2016). Conducted both in the laboratory and in the field, the aim of user testing is to collect feedback from users in order to identify usability problems or to obtain data about their experience of interacting with the software product (Bernhaupt, et al., 2016). In user testing, task analysis and task design are the main operations that reveal the functioning of the software system and specific user behavior (Hollnagel, 2012). In parallel to task analysis, use case analysis is a systematic approach to find out what users should be able to do with the software (Lethbridge & Laganiere, 2004). In use case analysis, users are described as actors and the interaction with the system is modeled by the roles of the actors and their actions (Lethbridge & Laganiere, 2004).

Task Analysis

The task is essential to the design of the user interface (UI); that is, the user's goal is to accomplish one or more tasks through the UI. To achieve this, the UI must provide mechanisms that allow the user to achieve his or her goal (Pressman, 2010). In usability evaluation, user testing focuses on these tasks. There are a number of success factors in conducting usability testing, the purpose of which is to accurately identify usability problems in the system. The characteristics of the participants, the test objectives, the task design, the problem criteria and the skills of the usability testers are among the factors that influence the success of the usability test (Lingaard & Chattratichart, 2007). In particular, task design is a critical activity in usability testing (McCloskey, 2021). In order to examine user interaction on the user interface design, task analysis is needed to direct user behavior and interaction. Use cases are a commonly used SE artifact for the specification of functional requirements, and the task models are used to capture the requirements and design information for the UI (Sinnig, et al., 2013).

In the course project carried out in this study, the task formulation of the usability test was carried out in parallel with the use-case modeling, thus aiming to encourage students to think about user behavior and to model full interaction within a scenario. Use case modeling via UML notations utilized in the analysis stage of the software development to represent how the user interacts with the system in order to complete a particular task, also shows interaction among users and system restrictions (Yue, et al., 2013). Use case diagrams also aim to specify how a system accomplishes each use case, and also provide high-level representation of what the system is designed to accomplish (Liaghati, et al., 2020). Modeling user involvement through the use case modeling addressed the question of how to describe and represent a particular task and how students can design the user interface to support that task (Constantine, 1995). By decomposing user tasks into components, Liaghati et al. (2020) suggested that a more detailed and measurable model would be helpful in modeling and measuring user behavior and system design. In this study, use-case modeling was carried out during the analysis phase and was intended to provide a road-map for students to see which interactions will be tested in the system during usability testing.

Efficiency, Effectiveness and Satisfaction

There are three usability variables that are evaluated in usability testing. Efficiency was measured by speed and interactivity, it related to the time taken by the user to complete a task, also the number of clicks made by a user to complete the task. Effectiveness is the task completion rate for each user, it shows the success rate for each task, it points to the number of errors per task (Georgsson & Stagger, 2016), while efficiency shows the effort rate for each user (Ferreira, et al., 2020). The third variable, satisfaction, shows user satisfaction and is measured by self-reported measurement tools. Including these three variables in the Usability assessment will affect the time and cost of the assessment (Ferreira, et al., 2020) and therefore also affect the UEP. Among these criteria introduced by the International Standardization Organization (ISO), satisfaction is freedom from discomfort and positive attitudes towards the use of the product (ISO 9241-11, 1998). Satisfaction gives the self-reported assessment that reflects the subjective judgment of the user. As the systems tested in this study are under development, the satisfaction measurement was not addressed. In this study, project teams measured and reported task completion time and task completion status metrics related to usability testing. Task completion status indicates whether the task was completed completely or with some error, and is reported as positive (+), half plus (±) or 'successful'. Failure was also reported negatively (-) or as 'failed'.

Problem Validity

Hartson et al (2003) state that the ultimate goal of usability evaluation is to identify real problems, and correct problem detection provides the designer with the necessary input for iterative design. The detection of qualified problems is widely investigated in studies. Nielsen showed that the probability of finding a usability problem in a test session can be approximated by a Poisson model (Nielsen & Launder, 1993). In addition, the severity of problems differs in terms of their probability of detection; severe problems are more likely to be found than less severe problems in both heuristic evaluation and user testing (Nielsen & Launder, 1993). Nevertheless, Nielsen and Landauer's mathematical model provides a formulation for the optimal amount of evaluation. In order to provide a more reliable and valid basis for usability evaluation, UEM studies have proposed criteria such as validity, thoroughness and efficiency; but these criteria focus only on the obtained problem set and do not provide guidance for other stages of the usability study. A study by Koutsabasis et al. (2007) compared different usability techniques in terms of the degree of realness of the problems found, but this does not adequately examine an evaluation method in terms of the attributes it should have in itself.

Studies show the impact of validity on usability findings, but no information is reported on how these findings are processed for. This study has attempted to show how the usability evaluation criteria, which have been shown to be effective, can be used to improve the design in the development process. The listed factors influence the success of the usability test on certain levels. In addition, usability evaluation methods (UEM) focus on some criteria to ensure the reliability and validity of the usability test results (Hartson, et al., 2003). When it comes to prototype design, UEMs focus on identifying and correcting usability problems before the final design through formative evaluation (Hartson, et al., 2003). In this study, validity, which is the proportion of real problems to the total usability findings, was measured and analyzed for the relationship between the number of solutions to improve the design.

Aim of the Study

The main objective of this study is to determine the relationship between usability evaluation performance and the number of problem solutions for design improvement. Secondarily, the study examines how students adopt the usability test results and lessons learned. The relationship between usability evaluation performance and the number of problem solutions was examined through correlation analysis. The relationship between the variables considered in the first and second research questions is shown in Figure 1. To answer the last research question, the reflection reports of the project teams were qualitatively analyzed to reveal the students' conclusions about the usability problem solutions.

RQ1: What is the usability evaluation performance of the project teams?

RQ2: Is there a relationship between usability evaluation criteria and the problem solving rate?

RQ3: How do students adopt the results of usability testing?

Significance

This study looked at some usability evaluation criteria in software development and showed the impact of these criteria on problem solving rate in terms of their contribution to design improvement. The SE course project integrated usability into the prototype development phase, focused on modeling user interaction and enabled students to adopt user-centered design. The study also analyzed students' reflections on design improvements with some of the coding examples. Throughout the project, students were encouraged to improve their designs by solving usability problems in a PBL approach. They were enabled to see and solve problems in the software development process. Thus, problem solving skills based on usability outcomes in software design were discussed by supporting the idea with qualitative data analysis.

2. Method

This study, designed as a case study, is based entirely on qualitative data. Document analysis and focus group interviews were used to collect data. The study focuses on software projects carried out by students in two different course semesters. Data sources of the study were students' course project reports and documentation, source codes of the projects on GitHub pages, usability test reports and final reflection reports, and interview data. Analyzed data were quantified and proportional values were obtained to address the usability evaluation criteria. Quantified data were analyzed using correlation analysis. Since the sample size was relatively small and it did not meet the assumption of normal distribution, Spearman Brown correlation analysis was utilized to reveal the relationship between UEP and the number of problem solutions (Myers & Sirois, 2006). To analyze problem solving behavior, content analysis was used to identify codes and themes. To analyze students' final conclusions, final reflection reports were examined and focus group interviews with project teams were conducted.

Study Group and Data Collection Process

The usability evaluators in this study were undergraduate students on the SE course in the Department of Management Information Systems. The students attended the course in their third year, after taking courses in programming and algorithm development and object-oriented programming in their first and second years. In the SE course, students applied the SDLC process by collaboratively developing software prototypes as part of the course project. Adopting an object-oriented approach, the project teams used the Unified Modeling Language (UML) to model the classes and objects of each project, after which they discussed how to include the user in the system through a use case scenario and model.

The number of cases in this study was 12 (twelve) software projects. Each project was carried out in groups of two or three people. The data were collected from the records and reports of the software projects of the students attending the Software Engineering (SE) course. All project documents (n=12) during two course periods (usability test reports, video recordings, source codes and final reflection papers) were examined. At the same time, the individual reflections of the students (n=30) in the project groups were analyzed through document analysis.

Data Analysis

Analysis of quantified criteria: Records of the usability tests conducted by the project teams (reports and video records) include the following information: task completion time, task completion status, user errors and problem list. In analyzing the usability evaluation criteria, this formulation was obtained from reports and videos through content analysis and was used as specified in the equations (EQ1, 2 and 3). To measure validity, the severity rate of each problem was rated qualitatively according to Nielsen's table.

Document Analysis: To answer RQ1 and partly RQ3, project documents, i.e., usability test reports, source code and documentation on GitHub project pages, and individual final reflection reports were analyzed using document analysis techniques. Some project teams provided video recordings they made during the usability tests. The video recordings were also analyzed to measure the severity of usability problems. Project source codes were examined to demonstrate problem solving behavior with the programming language used, quotes were also provided to show how they solve a particular problem. Final reflection reports were also exam papers of the students that they took a course grade, the reports were also analyzed through the related technique.

Thematic analysis: Interview sessions began with open-ended questions and continued with probes to elaborate on the responses. Thematic analysis was used to identify codes and patterns within the responses.

Project Development Process

In the SE course, students applied the SDLC process by collaboratively developing prototypes as part of the course project. Adopting an object-oriented approach, the project teams used the Unified Modeling Language (UML) to model the classes and objects of each project, after which they discussed how to involve the user in the system through a use case scenario and model at the requirement analysis. The model and scenario of only one of the projects are included as an example. Figure 1a shows the use-case model and Figure 1b shows the use-case scenario of the car rental project (P#3).

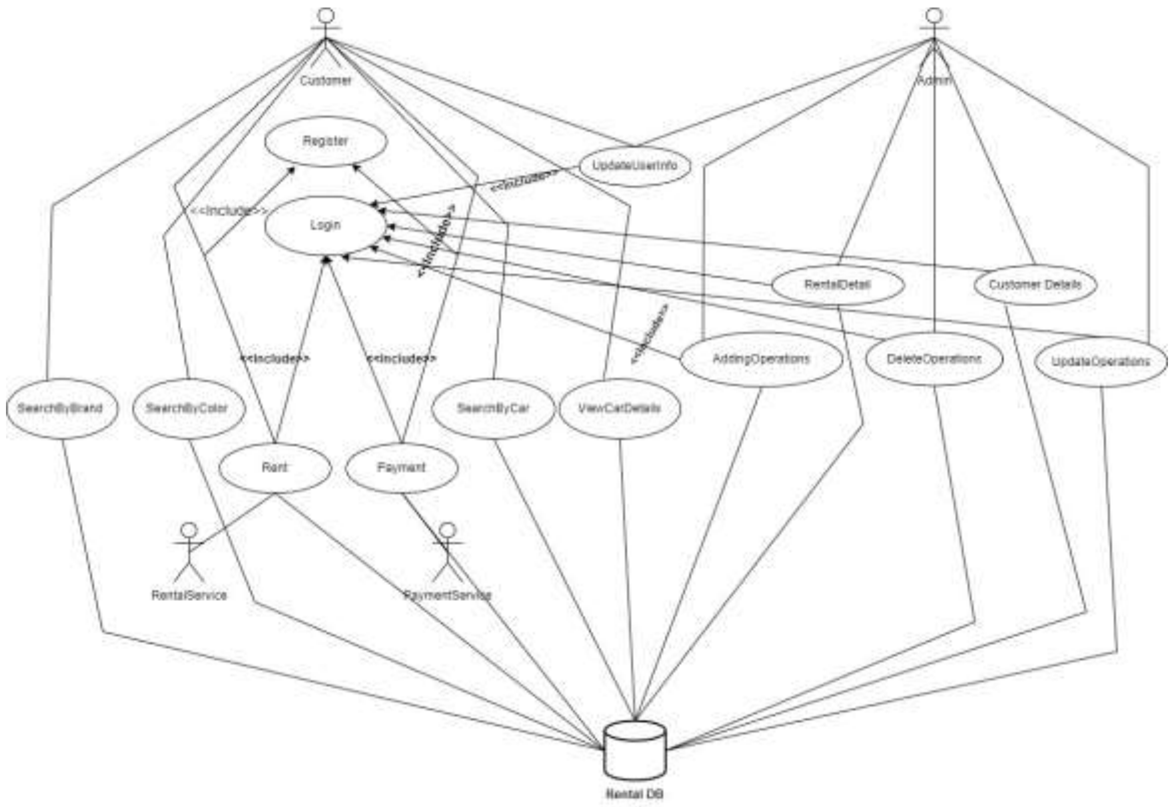
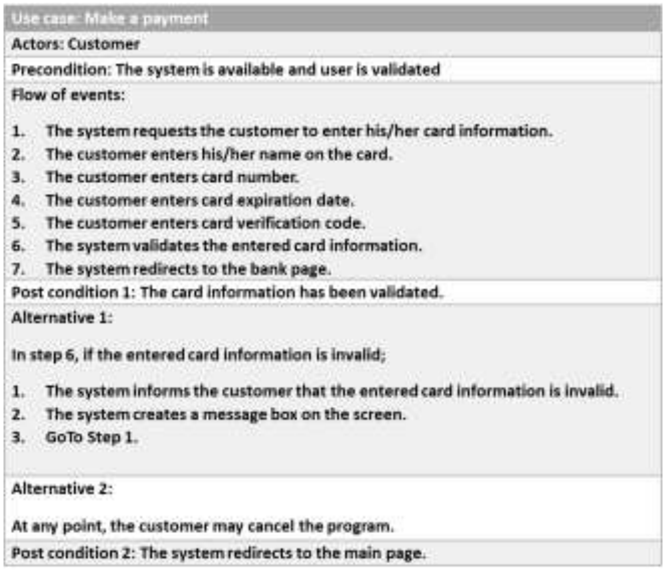


Figure 1a. Use case model of the car rental project



Use case: Make a payment

Actors: Customer

Precondition: The system is available and user is validated

Flow of events:

1. The system requests the customer to enter his/her card information.
2. The customer enters his/her name on the card.
3. The customer enters card number.
4. The customer enters card expiration date.
5. The customer enters card verification code.
6. The system validates the entered card information.
7. The system redirects to the bank page.

Post condition 1: The card information has been validated.

Alternative 1:

In step 6, if the entered card information is invalid;

1. The system informs the customer that the entered card information is invalid.
2. The system creates a message box on the screen.
3. GoTo Step 1.

Alternative 2:

At any point, the customer may cancel the program.

Post condition 2: The system redirects to the main page.

Figure 1b. Use case scenario of the car rental project

In Figure 1a there are two main types of users. There are main functions and the extensions were each described as a use case (Login, SearchByItem, ViewCarDetails), and other use cases connected to them with preconditions (Rent, Payment) for the customer, they also pointed out as "inclusions". There are also some drawbacks to the use case model in

Figure 1a; according to the scenario (Figure 1b), there is an alternative for the 'Payment' use case (Alternative 1), but this is not shown as an 'extension' in the model.

Task Design: Task design is the specification of tasks that reveal system-user interaction. Evaluators need to formulate the usability task in terms of observable behavior that shows system-user interaction. Table 1 shows the usability task list of the car rental project (P#3):

Table 1. Usability task list of the car rental project (P#3)

Task no	Usability task	Related page	Use case
1	Open the "Car Rental" home page	Home page	-
2	Filter "Peugeot" brand vehicles	Homepage	SearchByBrand
3	Filter vehicles with gray color	Homepage	SearchByColor
4	Register to the system required information	Register	Register
5	Log into the system via the mail and password	Login	Login
6	View details of the Volvo S60	Details	ViewCarDetails
7	Rent a car named Volvo S60	Rental	Rent
8	Make a payment	Payment	Payment
9	View the cars on rent	Admin page	RentalDetail
10	Add the car	Admin page	Adding Operations
11	Edit user information	User page	UpdateOperations

Table 1 shows the task list of P#3. Since the task list is created according to the use case model in Figure 2a, the use case that triggers the task is given. Accordingly, usability tasks are expected to test different components in the system, that is, to try out different usage scenarios. In the SE course, the students were informed about the methods of user interaction and usability evaluation of software, and they examined sample usability scenarios and solutions. They also drew several use case models of given scenarios. The SE course emphasized the integration of user modeling and user interaction into the design. In order to achieve this, the students were informed about the literature on human-computer interaction and user experience trends that serve the commercial value of the developed systems, thus the aim of the course was to ensure that the students adopt user-centered design.

GUI design: The project teams carried out the following activities in summary:

1. Conducting requirements analysis: an initial discussion session was held to determine user requirements, project teams wrote the initial set of use cases, and specified the software requirements they planned to develop.
2. Use case analysis and modeling was carried out to show planned interactions.

3. Project teams designed graphical user interfaces using several GUI design tools (PyQt5, Tkinter library for Python, and Visual Studio for Python and C# languages).
4. Usability tasks were formulated according to the use case models.
5. Usability tests were carried out with users.
6. Usability problems were solved, designs were reviewed.
7. Prototype versions were submitted.

Usability Evaluation Method

Usability evaluation through user participation concentrates on the testing of the system with real users. In order to carry out this technique, at least one working prototype of the system must be available (Dix, et al., 2004). In this study, the versions of the software projects before the usability test are called prototypes, and the projects whose errors are eliminated according to the results of the usability evaluation are accepted as full versions. Background information about usability testing is presented in Table 2:

Table 2. Background information about usability testing of teams

No	Project	Num. of users	Num. of tasks	Test environment	Total problems
P#1	Hospital information system	4	7	City hospital	3
P#2	Supermarket automation	6	5	Shopping center	3
P#3	Car rental	5	11	Rental office	7
P#4	Hospital appointment system	4	3	Uni campus	3
P#5	Activity tracker	3	7	Uni campus	6
P#6	Staff wanted	3	5	Shopping center	6
P#7	Refectory automation	3	6	Uni campus	3
P#8	Employee information system	4	5	Accounting office	6
P#9	Student information system	6	8	Uni campus	6
P#10	Dormitory student tracker	4	3	Uni Dormitory	4
P#11	E-commerce application	3	6	Uni campus	4
P#12	Stock control	3	8	Uni campus	5

Table 1 shows the general information about the usability studies of the teams, initially analyzed from the usability reports of the projects. According to Table 1, the evaluators carried out usability tests with different numbers of users. The usability test of the first project (hospital information system, P#1) was carried out in the city hospital with medical staff who are in charge of the information desk and are expected to use hospital automation tasks (add, delete and view patients, list appointments, etc.). Supermarket Automation Project (P#2) stores and manages the records and processes about the products of any supermarket. The car rental project (P#3) also keeps the information about cars in the database and the interface of the project helps to manage activities to add, remove, compare records and rent the cars. The Hospital Appointment System (P#4) is a web application that collects the data of hospitals and polyclinics and shows the availability for the selected date. The Activity Tracker project (P#5) stores and announces cultural and social activities. The application also allows users to attend the activity and save it in their calendar. The staff wanted project (P#6) is similar to the activity tracker, but it is a web application that brings together those who work in a subject and those who are looking for services and makes announcements. The refectory automation tool (P#7) has been evaluated for its functions related to the reservation of the meal list from a database. Other projects carry out several user tasks via the graphical user interface and they record user information and/or product information in the relational database.

During usability testing, the project teams used screen capture software and mobile phones to record user performance. To gain insight into user behavior, they used the think-aloud technique throughout the test sessions. The project teams also reported different usability metrics, state of fulfillment, task completion time or both. They also reported different numbers of total problems, ranging from 3 to 8.

Addressing Usability Evaluation Criteria

Efficiency and Effectiveness: Since each of the usability tests conducted by the project teams involved a different number of users and a different number of tasks, it is necessary to compare their usability evaluation performance in terms of these two variables. For this purpose, efficiency values were calculated for each user tested and effectiveness values for each task tested. As prototypes were used during the test and the main objective of the test was to identify critical problems, no satisfaction survey was carried out. In calculating efficiency in this study, the task completion time recorded by all project teams was used. Accordingly, the average time spent by all users on each task was summed and then the resulting value was divided by the total number of tasks in the test:

$$\text{Efficiency}=\frac{\sum Avg(task\ comp\ .time\ for\ each\ task)}{Total\ num\ .of\ tasks} \qquad \text{(EQ 1)}$$

Thus, obtaining the average time spent on each task. Similarly, effectiveness was calculated as the successfully completed tasks for each user, that is, as in the following equation was used as in the study of Fereira, et al., (2020):

$$Effectiveness = \frac{Avg\,(num.of\ successfully\ completed\ tasks)}{Total\ Sum.of\ tasks} * 100\%\ \text{(EQ 2)}$$

The effectiveness showed the percentage of success rate and it was calculated by taking the ratio of the average number of successful tasks by users to the total number of tasks. Situations other than "failure" contributed to the effectiveness calculation.

Problem Validity: Formative evaluation focuses on usability problems that need to be solved during the prototype design phase before a final design can be accepted for release (Hartson, et al., 2003). Validity is the measure that shows the proportion of problems that are real usability problems. It also shows the accuracy of the test or method, distinguishing between true and false alarms (Hartson, et al., 2003; Koutsabasis, et al., 2007). The validity was calculated using the following formula:

$$\text{Validity} = \frac{Num.of\ real\ problems\ found}{Total\ Findings\ As\ 'Problem'} \qquad \text{(EQ 3)}$$

Severity shows the quality of usability problems found by an evaluation method, and offers a guide for practitioners in deciding which usability problems are most important to fix (Hartson, et al., 2003). The "realness" value in the formula was suggested to be measured by "*severity level*" by Hartson et al. (2003). The degree of severity, which is the focal point of usability evaluation, is a criterion decided by the expert or practitioner and contributes to the development of the interaction design (Hartson, et al., 2003). In the study, the problems identified by the project teams during the usability test were scored between 0-4 according to Nielsen's severity rating scale as follows:

0: I don't agree that this is a usability problem at all
1: Cosmetic problem only: need not be fixed unless extra time is available on project
2: Minor usability problem: fixing this should be given low priority
3: Major usability problem: important to fix, so should be given high priority
4: Usability catastrophe: imperative to fix this before product can be released

Nielsen (1994) defined the severity of a usability problem using three factors: the frequency of occurrence of the problem, the impact of the problem on the user (will it be easy or difficult for users to overcome?) and the persistence of the problem (is it a one-off problem that users can overcome once they know about it, or will users be bothered by the problem repeatedly?) In EQ 1, usability problems that are above 2 (minor usability problem) according to the severity rating scale contribute to the number of real problems, while all problems scaled from 0 to 4 contribute to the total number of findings as 'problems'. A total of 67 problems were identified in the usability evaluations of 12 projects. The identified problems were rated by the researcher according to the severity scale. To ensure the reliability of the scoring, the help of two different coders was used. Instead of the " mean of a set of ratings from three evaluators " suggested by Nielsen, in this study, due to the high number of total problems, the problems were divided between two different coders. Accordingly, the problem set of 7 project teams was sent to a domain expert for evaluation, while an experienced front-end designer was used to score the usability problem set of 5 projects. While the problem sets were sent to the coders, the problem statement expressed by the team was sent directly or prepared in the context of the test scenario as shown in Table 4 under the relevant task heading. In the severity scoring used in the problem validity calculation, the average severity scores of the researcher and the second coder for the problem sets (n=31) belonging to 7 project teams and the average scores of the researcher and the third coder for 5 project teams (n=36) were obtained.

Examples of severity ratings are given in Table 3. The examples were presented within the usability testing scenario:

Table 3. Examples for severity ranking of usability problems

Type	**Example**	**Problem type**	**Problem Severity**
Task example of the P#3:	Filter vehicles with gray color/Volvo brand	Page design and navigation	4
System response:	Lists selected items at Homepage		
User feedback:	*"I can't refresh the search filter!"*		
Project team feedback:	*"Inability to reset the search form"*		
Task example of the P#3:	Rent a car named Volvo S60	Algorithm design	4
System response:	Show rental details for the selected item		
User feedback:	Asks to extend the rental period.		
Project team feedback:	*"inability to extend the rental period."*		
Task example of the P#4:	Register to the system by filling the form	Content design	3
System response:	*"Your registration is successful (Message-box)"*		
User feedback:	*"...too much time to fill this form!"*		
Project team feedback:	*"There are unnecessary information in the registration form, this could also affect the database performance"*		
Task example of the P#7:	Make a reservation for date of 2023-05-31, and save it	Error handling	4
System response:	*"Sorry, there is no specified day! (Message-box)"*		
Project team feedback:	*"wrong array definition of the days"*		
Task example of the P#12:	Open the sold items and income menu		
System response:	Displays information of sold products and calculates their income		
User-1 feedback:	*"It doesn't give me the name of the manufacturer; I would need it."*	Content design	2
User-2 feedback:	*"I get an error when viewing the sales details!"*	DB error	4

Table 3 shows the examples of severity ranking of usability problems and the types of problems in the context of usability test reports from teams. Examples are user responses during the interaction, project teams reveal the usability problem with the help of the detailed reports and video records. Problem severity value shows the mean of the severity ranking between raters.

Final Reflection

The reflections about the project were examined in two ways: individual and group reflections. The individual reflection reports written by the students in the final exam at the end of each semester were analyzed using the document analysis method. In the final exam the students were asked: "What would you do to turn your prototype into the final design? Through this question, the aim was to get the students' conclusions about the project and also to reveal what they had learned about the software development process.

Focus group interviews were conducted with the project teams to test their conclusions as a team. The focus group interviews were conducted by the researcher (who is also the course trainer) and detailed notes were taken throughout the discussions. In these sessions the groups were asked "What lessons have you learned from the project?" and "What is your next plan to complete your project? The aim was to find an answer on which all team members could agree. No audio recording was made, for reasons including the difficulty of sorting out informal conversations and ensuring that students felt comfortable during the sessions. In order to take efficient interview notes, one member of each group was appointed as a reporter, and by comparing the reporter's notes with the researcher's notes, shortcomings and unclear points were eliminated.

Reliability, Validity and Limitations

The study used a multifaceted data collection process to arrive at the conclusion determined by the third research question. This triangulation attempted to better capture and explain the complexity of human behavior (Cohen & Manion, 1989), that is, how students adopt usability test results. By using document analysis of usability test reports and project documentation on GitHub pages, more evidence of problem solving behavior could be gathered. Data collected through both individual reflection reports and focus group interviews provided two-way evidence of students' inferences about the design process. To provide further evidence, direct quotes were embedded in the findings. Due to the severity ranking table used in the validity measurement reflecting a subjective judgment, the quotations of the severity scoring are exemplified in their own context and presented in Table 4, thus aiming for the reader to be able to compare problems with different degrees of severity. The main limitation of the study is that 12 projects with usability evaluation measurements tested different systems on different users. Therefore, the measurements are independent of each other. Effectiveness and problem validity represented with proportional values, thus the correlation analysis performed on these values allows a rough judgment to be made.

3. Results

Correlation between Usability Evaluation Criteria and Problem Solutions

Analysis of the usability reports and recordings presented in Table 4 shows the task design and problem validity in percentage terms.

Table 4. Scores of the teams according to the usability evaluation criteria

	Num.of completed tasks	Effectiveness (%)	Efficiency (min)	Num.of severe problem	Validity (%)	Num. of solutions	Prob.solving rate (%)
P#1	6	85.7	6.40	1	33.3	2	66.6
P#2	4	80.0	1.76	2	66.6	2	66.6
P#3	9	81.8	3.55	6	85.7	5	71.4
P#4	2	66.6	4.46	1	33.3	1	33.3
P#5	5	71.4	2.20	2	33.3	3	50.0
P#6	4	80.0	3.34	1	16.6	2	33.3
P#7	5	83.3	6.38	1	33.3	1	33.3
P#8	2	40.0	5.21	2	33.3	1	16.6
P#9	7	87.5	2.54	5	83.3	4	66.6
P#10	1	33.3	3.74	1	25.0	1	25.0
P#11	5	83.3	1.80	2	50.0	2	50.0
P#12	6	75.0	4.00	1	20.0	2	40.0

The effectiveness of the usability evaluations was calculated as in the EQ 1, the presented values show the success rate of task completion status for projects. Accordingly, P#1, P#7, P#9 and P#11 have the highest effectiveness value, while P#8 and P#10 have the lowest effectiveness in terms of task success rate. Efficiency also shows the mean of the time per task as type of minute (as calculated by the EQ 2). A high value indicates that the average time spent on the task increases and therefore efficiency decreases. Respectively P#1, P#7 and P#8 have the lowest efficiency value. Groups with the highest efficiency are respectively P#2, P#5 and P#11.

The validity of the usability problems was calculated in relation to EQ 3; the proportion of real problems to the total number of findings was reported. P#3 and P#9 have the highest validity value (85.7% and 83.3% respectively), while P#6 has the lowest value (%16.6). The project teams varied between 16.6% and 71.4% in terms of their problem solving rates. P#8 solved 16.6% of the usability problems they identified, P#3 solved the 71.4%, while P#1, P#2, and P#9 solved 66.6% of the identified problems within the course period. P#8 have relatively low effectiveness and efficiency values which have the minimum problem solving rate. P#9 also has a high effectiveness value and the problem solving rate.

In order to analyze the correlation between usability evaluation criteria and problem solving rate, Spearman-Brown correlations were calculated. The result showed that there was a positive correlation between effectiveness and the problem solving rate (rho=.679; p=.008), validity and the problem solving rate (rho=.704; p=.005). Efficiency is the average completion time in minutes. Percentiles of efficiency have been calculated to examine the correlation between this score and problem solving rate. The correlation between efficiency and the problem solving rate was negative but not significant (rho=-.437, p=.078).

Problem Solving Behavior

Based on the usability test results, the project teams fixed the bugs in their project source code. The teams completed the prototype designs by fixing the bugs before submitting their projects. The problem solving behavior was examined by analyzing the usability test reports and project source codes on GitHub pages, the results are shown in Table 5 and 6:

Table 5. List of solved problems

Themes	*n*
Page design and navigation problem -button not working -redirect to wrong page	5
Error handling -value error exception (error message for integer value) -missing character-length definition while user register -wrong exception defining	5
Database (DB) error -DB connection error -wrong query design	2
Algorithm design problem -conditional loop error -improper function definition -lack of function definition	6

Search matching error -keyword defining & indexing in DB	3
System-user communication error: -system dialog object design	1
Other GUI problems -pop-up menu design -window size, font size&color	3
Content design	2
Total	26

Table 5 shows the solved problems and their categories in terms of programming context. The page design and navigation problems (n=5) are related to whether the navigation and redirects between pages work correctly in the system. Five navigation problems were identified and solved during the usability test. Error handling problems are related to whether the system informs the user correctly and with appropriate objects (n=5). In contrast, the correct understanding and interpretation of user input by the system indicates a satisfactory keyword matching during the search (n=3). The correct keyword definition was made in response to the identified problem. Problems that arose during the addition of data to the system or the retrieval of recorded data led the designers to check the database connection (n=2). Other fixes relate to the system's algorithm design (n=6), language syntax errors and initialization of GUI elements (n=5). Source code examples of the fixed problems are also given in Table 6.

Table 6. Some usability problems and solutions

Usability problem	*Source of the problem*	*Solution*

Back button in the program did not work normally (P#1)	Page design & navigation problem:
	Button not working
		in-page menu was created in Python:
		
		self.ui.aboutMenu.triggered.connect(self.about)
		self.about_page = About()
		--- 	
		def about(self):
		self.about_page.show()

Trouble in removing product in P#2, user could not get feedback about transaction	System-user communication error:
	Lack of system dialog object definition
		Exception defining: the product entity was queried before being removed, and the status was printed in Python:
		
		if len(product_list)==0):
		print(“there is no product to delete”)
		else:
		print(“the product has been deleted”)

inability to extend the rental period (P#3)	Algorithm design error	Defining class that contain required method in C#:
		…
		CheckRentDate(Rental rental){
		var
		result=_rentACarDal.GetAll(r=>r.CarId==rental.CarId
		&&(r.RentDate<=rental.RentDate&&rental.RentDate<=
		r.ReturnDate) || (rental.RentDate <=r.RentDate &&
		r.RentDate<=rental.ReturnDate));
		if (result.Count==0){return new SuccessResult();}

		else
		{return new ErrorResult();}

User had difficulty in finding the department, (P#4)	Search matching error: No match with keywords of user and the system	Keywords were redefined and indexed for search in the relational DB.

Trouble in reservation for available day in P#7	Error Handling	Array re-definition for days of the months in C#:
		
		int dayno, cntr =0, empty = 31, full = 0;
		int[] arrayfullday = new int[0];
		…
		if (dayno < 1 || dayno > 31){
		MessageBox.Show("Please enter a valid day no!", "Warning", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);…}

Trouble in add, remove and display records in P#9	DB connection error	Rewrite the database connection in C#:
		
		private void button1_Click(object sender, EventArgs e){
		connection.Open();
		SqlCommand save = new SqlCommand("insert into student(…

Table 6 shows the examples of coding for the problem solving behavior of some of the projects. In the example of the hospital information system (P#1), the team noticed that some of the back buttons did not work or went to the wrong place. The team described the problem in their report as follows: *"...the back button in the program does not work normally, when they want to go back, they press the cross..."*. They explained that the reason the button did not work was due to the class definition of the GUI element; instead, they designed navigation through pages, as shown in Table 6.

In P#2, the team reported that two market cashiers had difficulty removing the product because the systems did not inform the users, so they were confused about the result of the transaction. The team solved this problem by adding an if-else statement to check the presence of the product in the DB. They also found that relatively older users had problems with the interface, they noted the suggestion to simplify the interface design, that is, make buttons more visible and add info-graphics associated with system functions on the GUI.

In the example of P#3, the team defined the acquired method in a class structure *"...in order to easily track the errors we received during testing, because we had created the GUI in an abstract base class, so the GUI elements were designed in a hierarchy and the tasks were distributed to classes...we defined a class that contained the method of extending the rental period, under the base class, clean job!"*

In the example of P#4, the problem arose from an unfinished DB design; the team reported the problem as *"...some users wrote keywords that we had not yet registered in our system at the last stage, as a result there was no match with any hospital or department*". The team addressed this issue first and foremost: *"…we added more keywords and indexed by keyword in the DB"*.

In the example of P#7, the team corrected the array definition for the days of the month to allow the user to select an available day for reservation *"...when she tried to make a reservation using the save button, she got an error: "Sorry, there is no day specified!*". This was due to the code; the array of days was defined so that each month was a 30-day period.

In the example of database connection errors (P#9), the team expressed that the problem originated in their source code; they checked the database connection commands and fixed the problem. On the other hand, the team reported user-generated problems arising from database operations; some users had difficulty displaying records; they redesigned the query forms on the screen and added footers to help users with their queries. After solving the specific problems listed in Table 5, the whole team updated the user guide for topics where users needed help during interaction. P#1 expressed: *"...after the usability test, we tried to develop a more user-friendly interface, we removed redundant elements that caused confusion and we updated our user guide"*.

Results on Final Conclusions

By analyzing the individual reflection reports and focus group interviews, the students' conclusions about the project and their future plans were obtained.

Individual reflections: A total of 30 individual reflection reports were analyzed through document analysis and 23 codes were found. The results are categorized into the following themes in Table 7:

Table 7. Individual reflections for future improvements

Themes	*Codes*	*n*
GUI design improvements	Add new elements to the interface	3
	To simplify interface	3
	Develop mobile version of program	3
	More responsive design	2
Algorithm design improvements	Define more system dialog object to inform user for critical tasks (i.e., stock control)	1
	Rebuild the algorithm to handling exceptions	1
	Using design patterns to design more hierarchical way	1
	Make more automation test before user testing	2

Database design improvements	Indexing stock information with serial number of products	1
	Write trigger to control DB updates	1
	User authorization on DB (Grant privileges)	2
	Maintain and backup DB	1
Extensions	Add payment module to dormitory student tracker	1
	Add a module to analyze user data with machine learning and recommends appropriate appointment information	1
Total		23

Table 7 shows the students' plans for future improvements. The most frequently expressed plan was to improve the GUI design; to extend or simplify the user interface or to make the design more compatible for mobile or desktop. In the category of algorithm design improvements, students planned to improve system-user interaction (i.e., catching errors and informing the user) using error handling mechanisms. DB design improvements include plans to speed up DB access and increase DB security. Extensions include the new modules that the students plan to add to their project in the future.

Focus group interviews: In the focus group interviews, project teams were asked about lessons learned from the project process, including their next plans and reflections as a team. They were also asked about the challenges during the usability testing. Table 8 shows the findings on lessons learned:

Table 8. Lessons learned from the projects

Themes	*Codes*
Importance of algorithm design	Pseudo codes and drafts help to improve design
	Algorithm is the most important part of the project
	"The user" should be considered both in back-end and front-end
Benefits of user-centered design	Understand what the user needs
	Looking at the design from the user's eye
	Imagine the user as a customer in the future
	Developing human resources and marketing skills
Model-based design	Scheduling and organization of steps with the help of SDLC
	Looking at the big picture thanks to modeling
Problem solving skill	Collaboratively analyze and solve problems
	More problems, more tips
	Learn from problems
Challenges	Recruiting users
	Conducting usability test
	Understand user
	Constraints arising prototype

As shown in Table 8, the first theme is the importance of algorithm design. The project teams expressed their opinions on algorithm design. They designed and improved the algorithm of their project until the final version of the prototypes, so they learned that the usability factor should be considered both in the back-end and in the front-end simultaneously. Benefits of user centered design, gives insight into developing understanding on user centered design. Project teams expressed that they understand what the user needs, looking at the design from the user's eye. Considering the user as a customer in the future, they learn about the customer relationships. The conclusions of one of the teams on this subject are as follows:

"…we realized that conducting usability testing and interacting with users one-on-one increases the likelihood that the project will be preferred over its competitors. Presenting and marketing a real project to customers in the future requires knowing UX terms."

"Designing interaction well and putting the user first will help us get the job!"

In the Model-based design category, teams were positive about the SDLC model. Here are some quotes on the topic of model-based design:

"The worst thing is not knowing where to start. SDLC gave us the opportunity to plan and schedule all the steps..."

"We would like to have a similar experience with different models and projects!"

"…we learn to put ourselves in the user's shoes while drawing the use-case model. …while doing the usability test, we realized that we wish we had written more alternative scenarios in our use-cases."

In the category of problem solving skills, they stated that they saw the benefits of working together to solve problems. They also stated that the problems that arose during usability testing not only showed them programming errors, but also gave them clues as to how to solve these errors. In this way they saw that the findings, called 'problems', helped to improve the design. Here are some quotes on the subject:

"The user showed us how many mistakes there were in our seemingly perfect project!"

"We solved problems much faster with work sharing."

In the challenges category, project teams mostly emphasized the difficulty of recruiting users for tests. Convincing the user to test was the most frequently expressed situation. The problems experienced by the user in devoting time to the tasks, finding a suitable environment for testing, getting the necessary permission for video recording and communicating with the user. Other problems identified were due to the limitations of the prototype.

"However, it was difficult to address user requests… the users' reaction is sometimes not realistic, sometimes even "inscrutable…it is crucial to contact the right people!"

"…It is also important to have a sufficient prototype… If the user is redirected to a blank page during the execution of the task, or if the user clicks on a button that we have not yet taken any action on, this can be a distraction for the user."

"It (usability test) is difficult to overcome without a team."

The project teams also drew some conclusions about their future plans. P#1 noted that they will complete the design functions and add elements to the GUI for new functions. P#2 planned to extend the project with a customer module and they expect that their project will be requested by some big supermarket chains in the city. P#4 project team planned to extend the user automation with a payment module in addition to solving the current problems in the university hostel. Other project teams planned to develop more user-friendly interfaces. They wanted to get involved in large projects and put into practice what they had learned in the course.

Conclusion

The results confirmed that high validity usability testing helps to produce more design solutions and helps to solve more problems. Project teams with high test validity produced more solutions to the problems they were given. When looking at problem solving behavior of the project teams, they solved the problems respecting navigation design, error handling, database connection, algorithm design, search matching, system-user communication and GUI design. Positive correlation between problem validity and problem solving rate also indicates that problems marked as "severe" carry more clues for design improvement. This increases the likelihood that such problems, which irritate users the most during testing, will be reported to the designers, thus encouraging them to create solutions. Another relationship was seen between effectiveness of usability evaluation and problem solving rate. The success rate regarding the task completion status was also correlated with the problem solving rate.

When the teams' conclusions for the final design were analyzed, it became clear that they were gaining valuable insights into user-centered design. Looking at the modifications pointed out in the reflection reports and the project source codes on GitHub, it seems that the project teams improved their prototypes by solving usability problems. It can also be concluded that the students received useful hints for the final design of an information system and a real project.

Looking at individual and focus group reflections, students have made significant gains in terms of software design as a result of the approach adopted in the course. In addition to improving the front-end and back-end software skills acquired in previous courses, the project-based application and user testing is considered useful in preparing them for real-life projects. Corrections made to the algorithm and database design in response to user feedback, as well as examples of front-end design improvements, could be taken as evidence of positive outcomes from the process.

Implications for SDLC

Usability is an important quality attribute for software applications and needs to be paid attention to throughout the development phase of the SDLC (Gupta, et al., 2017; Velmourougan, et al., 2014). The results of this study showed that as the usability evaluation performance increased, the problem resolution rate also increased. Consequently, it can be said that the results obtained from usability tests with high performance respecting criteria encourage students to solve more problems. This can be interpreted as they put more emphasis on the user during the design phase. Problem severity can be considered as an effective factor in solving usability problems. Identifying and solving such problems during the prototype design phase helped students to learn the importance of user-centered design by doing and experiencing. Ponce et al. (2018) also concluded that in order to achieve a successful interface design, critical usability problems need to be found and solved in the early stages of design, especially with regard to serious design issues. Another outcome of the study may be that students gain the ability to adapt the skills they have learned in programming courses to the needs of customers, thus facing real life problems. Both individual and group reflections support this conclusion. Therefore, it would be beneficial to have more integration between coding courses to give students the opportunity to apply what they have learned. Similar to this study, Segura (2021) carried out an undergraduate course on usability-integrated software development, and usability integration enabled students to develop prototypes of the system of higher quality and more usable. The author emphasized that the teaching of usability in software development in engineering education contributed to satisfactory results, so it needs to be included in the curriculum.

Discussion

This study makes a case for integrating usability into the SDLC; incorporating usability into any life-cycle model for prototype design, provided that problems are identified early and addressed throughout development. Although the user element is more prominent in agile processes (Pressman, 2010), it has been empirically proven that an efficient design process can be realized by integrating it into the traditional software development process. From the perspective of agile and traditional models that provide different approaches to the user element, integrating usability into the design process could make traditional models more flexible and iterative (Silva, et al., 2015). The model introduced by Velmourougan et al. (2014), which places usability at the center of SDLC, proposes to consider usability testing scenarios throughout software development. They expressed that the implementation of the proposed model will minimize user effort during interaction. In this study case, the final conclusions of the project teams showed the benefits of integrating usability at the early stage of design. They pointed out critical improvements intended for main components (GUI and algorithm design, database and extensions) of each user task automation. A similar case study conducted by Yoon, et al. (2017) also concluded that this type of interaction can be used to iteratively identify, remove and avoid potential problems in the development of user interface prototypes.

In this study, students conducted usability evaluation with user testing, so it is not clear that the effect of other usability evaluation methods on design problem solving behavior, i.e., heuristic evaluation. Maguire and Isherwood (2018) found user testing has the potential to detect more severe problems than heuristic evaluation; based on Nielsen's severity ranking measurement, the researchers found that user testing had a severity average of 2.02 while heuristic evaluation of 1.71. In the literature, there are efforts to examine usability evaluation methods in terms of performance and develop more objective measures (Hartson, et al., 2003). Unlike objective measurements, subjective usability assessments need to be verified with different measures and different evaluators (Hornbæk, 2006). In this study, it may be necessary to verify the scoring key that evaluates the course projects in terms of task formulation skills by testing them on different samples.

Usability was integrated into a software engineering course using a project-based and problem-solving approach. A positive correlation was found between students' performance in usability evaluations and their rate of solving software problems. At the same time, multidimensional qualitative analyses show that students had positive learning experiences. Observing real-life user behavior during usability testing not only helped students identify software errors, but also provided useful hints for future designs. Borys (2016) concluded that high fidelity prototyping and usability testing in software development lab experiences help students achieve positive learning outcomes, both in the course and in terms of future professional life. On the other hand, the author reported that formal documents such as usability reports are not interesting for students. In this study, usability reports are the basis for students to provide evidence for field testing and to evaluate test performance.

This study demonstrates the positive outcomes of collaborative problem solving on students' course experience in software engineering education. Ciancarini, et al. (2019) investigated collaborative thinking and its latent variables such as group awareness, group organization and complex negotiation in software development and concluded that solving complex software problems collaboratively in a team has a positive effect on problem solving skills, along with other variables considered (i.e. computational thinking).

This study also presented the importance of the quality of the usability evaluation, which positively complements the design process. In fact, real software projects do not perform user testing with real users although they know its importance due to the difficulty of integrating user testing in agile iterative processes (Silva, et al., 2015). In order to reduce the cost of user testing, alternative methods (i.e., A/B testing) have been proposed to apply refactoring and improve usability (Firmenich, et al., 2019). In this study, the integration of usability into the process, which is limited to training purposes and prototype development, can give practitioners an idea of its inclusion in real projects. It is assumed that students develop their ability to interpret the results of usability testing and reflect them in the interaction design solutions. It is expected that the skills they acquire will help them to adopt a user-centered approach in challenging work in the future.

Effectiveness and efficiency are variables that contribute to the performance of usability evaluation. Effectiveness, which is the rate of doing or accomplishing the task correctly, also contributes positively to usability evaluation performance. This situation also shows the importance of "conscious" users who can fulfil the task of an information system correctly and effectively. The expression "contact with the right people" among the findings of the focus group interviews supports this idea. Therefore, recruiting users who understand the system and can perform tasks correctly can contribute to improving the system. This could also increase the efficiency of usability testing. Yet there needs to be more data analysis to prove this. Georgsson and Staggers (2016) also proved the effect of user characteristics, with users having more experience with information technology and younger users showing higher performance in terms of effectiveness and efficiency in a usability test of a personal health support system.

Validity indicates the proportion of usability problems up to a certain severity level, but in accordance with the iterative design process, low severity problems in prototype design and even details that the designer noticed during testing are also taken into account in the final design (Hertzum, 2006). The positive correlation between the validity of the usability problems and the rate of problem solving suggests that severe problems help designers to produce more solutions, which could encourage designers to improve the design. A review of the literature shows that validity measurement is also used to compare different usability evaluation methods (Koutsabasis, et al., 2007; Maguire & Isherwood, 2018). In this study, it could be said that usability test results with high validity contributed to complete a prototype or incomplete software elements in terms of end-user feedback. On the other hand, situations related to the characteristics of the prototype (i.e. fidelity level) affect the attitudes and behavior of both the user and the evaluator during usability testing (Lim, et al., 2006). This shows that the criteria for usability measurement can also be influenced by the design style of the prototype. In this study, prototype-related variables are not discussed, it is assumed that all projects have sufficient functionality for usability testing separately. This can be seen as a limitation of the study.

Formulating usability tasks according to use case scenarios allowed students to think about design from the perspective of the end user; designing the software with usability in mind allowed students to grasp design thinking (Martins, et al., 2019). Incorporating use case scenarios and usability testing into prototype development could support user interaction for the finished version of the system, it could also eliminate the need for iteration and usability costs (Elkoutbi, et al., 2006). The more elaborate model proposed by Liaghatti (2020) at the prototype stage gives the opportunity to evaluate the system before testing it with real users, but for developers with insufficient field experience, user testing allows them to examine real user behavior on their designs. Learner programmers have to take the above considerations into account in the competitive environment of the software market. Through a systematic review of literature, Curcio et al. (2019) stated that agile processes applied in everyday life focus on software functionality and ignore user-centered design, and focusing on customer needs does not guarantee usability. Therefore, they conclude that the integration of usability into software development is still an emerging issue when it comes to user-centered design.

Ethical Committee Permission Information

Name of the board that carries out ethical assessment: Adana Alparslan Türkeş Bilim and Technology University Scientific Research and Publication Ethics Board

The date and number of the ethical assessment decision: 28.02.2022 -29675

Author Contribution Statement

Nihal MENZİ ÇETİN: *Conceptualization, methodology, data analysis, and writing.*

References

Allen, D.E., Donham, R.S. and Bernhardt, S.A. (2011), Problem-based learning. New Directions for Teaching and Learning, 2011, 21-29. https://doi.org/10.1002/tl.465

Alshamari, M. & Mayhew, P. (2008). Task design: Its impact on usability testing. *Proceedings of the third international conference on internet and web applications and services*, (pp. 583-589). https://doi.org/10.1109/ICIW.2008.20

Bernhaupt, R., Palanque, P., Manciet, F., Martinie, C. (2016). User-test results injection into task-based design process for the assessment and improvement of both usability and user experience. In C. Bogdan, et al. (Eds.), *Human-Centered and Error-Resilient Systems Development. HESSD HCSE 2016 2016. Lecture Notes in Computer Science*, (pp. 56-72). Springer, Cham. https://doi.org/10.1007/978-3-319-44902-9_5

Boehm, B. (2005). The future of software processes. In E. Bertino, W. Gao, B. Steffen & M. Yung (Eds.). *Lecture Notes in Computer Science, Software Process Workshop* (pp. 10-24). Springer Berlin Heidelberg.

Bolchini, D., Garzotto, F. (2007). Quality of web usability evaluation methods: An empirical study on MiLE+". In M. Weske, M. S. Hacid, C. Godart, (Eds.), *Web Information Systems Engineering – WISE 2007 Workshops*, LNCS (vol. 4832), (pp. 481-492).

Borys, M. (2016) Teaching software usability engineering: classroom experience, *INTED2016 Proceedings*, (pp. 2712-2717). http://doi.org/10.21125/inted.2016.1590

Brodie, L., Zhou, H., & Gibbons, A. (2008). Steps in developing an advanced software engineering course using problem based learning. *Engineering education, 3*(1), 2-12. https://doi.org/10.11120/ened.2008.03010002

Chattratichart, J., & Brodie, J. (2004, April). Applying user testing data to UEM performance metrics. In *CHI'04 extended abstracts on Human factors in computing systems* (pp. 1119-1122). https://doi.org/10.1145/985921.986003

Ciancarini, P., Missiroli, M., & Russo, D. (2019). Cooperative Thinking: Analyzing a new framework for software engineering education. *Journal of Systems and Software*, *157*, 110401. https://doi.org/10.1016/j.jss.2019.110401

Cico, O., Jaccheri, L., Nguyen-Duc, A., & Zhang, H. (2021). Exploring the intersection between software industry and Software Engineering education-A systematic mapping of Software Engineering Trends. *Journal of Systems and Software*, *172*, 110736.

Cohen, L., & Manion, L. (1989). *Research methods in education (Third edition).* Routledge.

Constantine, L. L. (1995). Essential modeling: Use cases for user interfaces. *Interactions*, *2*(2), 34-46.

Curcio, K., Santana, R., Reinehr, S., & Malucelli, A. (2019). Usability in agile software development: A tertiary study. *Computer Standards & Interfaces*, *64*, 61-77.

Dix, A., Finlay, J., Abowd, G. D., & Beale, R. (2004). *Human computer interaction (third edition)*. Pearson Prentice Hall.

Elkoutbi, M., Khriss, I., & Keller, R. K. (2006). Automated prototyping of user interfaces based on UML scenarios. *Automated Software Engineering*, *13*, 5-40.

Ferreira, J. M., Acuña, S. T., Dieste, O., Vegas, S., Santos, A., Rodriguez, F., & Juristo, N. (2020). Impact of usability mechanisms: An experiment on efficiency, effectiveness and user satisfaction. *Information and Software Technology*, *117*, 106195.

Firmenich, S., Garrido, A., Grigera, J., Rivero, J. M., & Rossi, G. (2019). Usability improvement through A/B testing and refactoring. *Software Quality Journal*, *27*, 203-240.

Fischer, H. (2012). Integrating usability engineering in the software development lifecycle based on international standards. In S. D. J. Barbosa, J. C. Campos, R. Kazman, P. Palanque, M. Harrison (Eds.), *Proceedings of the 4th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, (pp. 321-324). Association for Computing Machinery, New York. https://doi.org/10.1145/2305484.2305541

Georgsson, M., & Staggers, N. (2016). Quantifying usability: An evaluation of a diabetes mHealth system on effectiveness, efficiency, and satisfaction metrics with associated user characteristics. *Journal of the American Medical Informatics Association*, *23*(1), 5-11.

Ghezzi, C., & Mandrioli, D. (2005). The challenges of software engineering education. *In Proceedings of the 27th international conference on Software engineering (ICSE '05). Association for Computing Machinery, New York, NY, USA*, (pp.637–638).

Gulati, A., Dubey, S. K. (2012). Critical analysis on usability evaluation techniques. *International Journal of Engineering Science and Technology*, *4*(3), 990-997.

Gupta, D., Ahlawat, A., & Sagar, K. (2017). Usability prediction & ranking of SDLC models using fuzzy hierarchical usability model. Open Engineering, 7, 161-168.

Hartson, H. R., Andre, T. S., & Williges, R. C. (2003). Criteria for evaluating usability evaluation methods. *International Journal of Human-Computer Interaction*, *15*(1), 145-181.

Hertzum, M. (2006). Problem prioritization in usability evaluation: From severity assessments toward impact on design. *International Journal of Human-Computer Interaction*, *21*(2), 125-146. https://doi.org/10.1207/s15327590ijhc2102_2

Hollnagel, E. (2012). Task analysis: Why, what and how. In G. Salvendy (Ed.), *Handbook of human factors and ergonomics (fourth edition).* (pp.385-396). John Wiley & Sons, Inc.

Hornbæk, K. (2006). Current practice in measuring usability: Challenges to usability studies and research. *International Journal of Human-Computer Studies*, *64*(2), 79-102.

Koutsabasis, P., Spyrou, T. & Darzentas, J. (2007). Evaluating usability evaluation methods: criteria, method and a case study. In *Proceedings of International Conference on Human-Computer Interaction*, (pp. 569-578).

Kuhnel, M., Seiler, L., Honal, A., & Ifenthaler, D. (2018). Mobile learning analytics in higher education: Usability testing and evaluation of an app prototype, *Interactive Technology and Smart Education*, *15*(4), 332-347.

Lethbridge, T.C. & Laganiere, R. (2004). *Object-oriented software engineering. Practical software development using UML and Java (second edition)*. London: McGraw-Hill.

Liaghati, C., Mazuchi, T., & Sarkani, S. (2020). A method for the inclusion of human factors in system design via use case definition. *Human-Intelligent Systems Integration*, 2, 45–56.

Lim, Y. K., Pangam, A., Periyasami, S., & Aneja, S. (2006, October). Comparative analysis of high-and low-fidelity prototypes for more valid usability evaluations of mobile devices. In *Proceedings of the 4th Nordic conference on Human-computer interaction: changing roles* (pp. 291-300). https://doi.org/10.1145/1182475.1182506

Lingaard, G., & Chattratichart, J. (2007). Usability testing: What have we overlooked? In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, (pp. 1415-1425). San Jose, USA. https://doi.org/10.1145/1240624.1240839

Madan, A., & Dubey, S. K. (2012). Usability evaluation methods: A literature review. *International Journal of Engineering Science and Technology, 4*(2, February), 590-599.

Maguire, M., & Isherwood, P. (2018). A comparison of user testing and heuristic evaluation methods for identifying website usability problems. In *Design, User Experience, and Usability: Theory and Practice: 7th International Conference, DUXU 2018, Held as Part of HCI International 2018, Las Vegas, NV, USA, Proceedings, Part I 7* (pp. 429-438). Springer International Publishing.

Martins, H. F., Oliveira, A. C., Canedo, E. D., Kosloski, R. A. D., Paldês, R. A., & Oliveira, E. C. (2019). Design thinking: Challenges for software requirements elicitation. *Information*, *10*(12), 371. https://doi.org/10.3390/info10120371

McCloskey, M. (2021). Turn user goals into task scenarios for usability testing. https://www.nngroup.com/articles/task-scenarios-usability-testing/ 09.11.2021.

Mitchell, G. G., & Delaney, J. D. (2004). An assessment strategy to determine learning outcomes in a software engineering problem-based learning course. *International Journal of Engineering Education*, *20*(3), 494-502.

Myers, L. & Sirois, M. J. (2006). Spearman correlation coefficients, differences between. Encyclopedia of statistical sciences, 12. https://doi.org/10.1002/0471667196.ess5050.pub2

Nielsen, J. (1994). *Severity ratings for usability problems*. https://www.nngroup.com/articles/how-to-rate-the-severity-of-usability-problems/

Nielsen, J. & Landauer, T. K. (1993). A mathematical model of the finding of usability problems. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems,* (pp. 206-213). https://doi.org/10.1145/169059.169166

Ponce, P., Peffer, T., & Molina, A., (2018). Framework for evaluating usability problems: a case study low-cost interfaces for thermostats. *International Journal of Interactive Design and Manufacturing, 12*(2), 439-448. https://doi.org/10.1007/s12008-017-0392-1

Pressman, R. S. (2010). *Software engineering. A practitioner's approach (seventh edition)*. McGraw Hill International Edition.

Puerta, A. R. (1996). The Mecano Project: Enabling user-task automation during interface development. In *AAAI Technical Report (AAAI, 1996)*, (vol. 96, pp. 117-121).

Segura, J. (2021). The teaching of usability in software development: Case study in the computer engineering career at the university of Matanzas. *International Journal of Engineering Pedagogy, 11*(1), 4-15.

Silva, T. S., Silveira, M. S., & Maurer, F. (2015, January). Usability evaluation practices within agile development. In *2015 48th Hawaii International Conference on System Sciences* (pp. 5133-5142). IEEE. https://doi.org/10.1109/HICSS.2015.607

Sinnig, D., Chalin, P., & Khendek, F. (2008). Consistency between task models and use cases. In *Engineering Interactive Systems: EIS 2007 Joint Working Conferences, EHCI 2007, DSV-IS 2007, HCSE 2007, Salamanca, Spain,* (pp. 71-88). Springer Berlin Heidelberg.

URL: J. Nielsen, Why you only need to test with 5 users?. https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/ 09.09.2021.

Velmourougan, S., Dhavachelvan, P., Baskaran, R., & Ravikumar, B. (2014, September). Software development Life cycle model to build software applications with usability. In *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (pp. 271-276). IEEE.

Yoon, H., Park, S. H., Lee, K. T., Park, J. W., Dey, A. K., & Kim, S. (2017). A case study on iteratively assessing and enhancing wearable user interface prototypes. *Symmetry*, *9*(7), 114. https://doi.org/10.3390/sym9070114

Yue, T., Briand, L. C., & Labiche, Y. (2013). Facilitating the transition from use case models to analysis models: Approach and experiments. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, *22*(1), 1-38. https://doi.org/10.1145/2430536.2430539

Zainol, A. & Almukadi, W. S. (2020). Implementing problem-based learning in the software engineering course. *International Journal of Advanced and Applied Sciences*, *7*(12), 19-26. https://doi.org/10.21833/ijaas.2020.12.002

Copyright © JCER

JCER's Publication Ethics and Publication Malpractice Statement are based, in large part, on the guidelines and standards developed by the Committee on Publication Ethics (COPE). This article is available under Creative Commons CC-BY 4.0 license (https://creativecommons.org/licenses/by/4.0/)