Topic: Human-Computer Interaction, Computer Systems

Development and validation of a Descriptive Cognitive Model for predicting usability issues in a Low Code Development Platform

Carlos Silva[1], Joana Vieira[1], José C. Campos[2,3], Rui Couto[2,3], António N. Ribeiro[2,3]

[1]Center for Computer Graphics, Guimarães, Portugal

[2]Department of Informatics, University of Minho, Braga, Portugal

[3]HASLab/INESC TEC, Braga, Portugal

Corresponding author: Rui Couto, rmscouto@gmail.com

Word count: 9468 (text) + 1010 (references)

Manuscript type: Special Section Article

PRÉCIS

This study proposes and evaluates a Descriptive Cognitive Model (DCM) for the identification of initial usability issues in a low-code development platform (LCDP). By applying the proposed DCM we were able to predict the interaction problems felt by first-time users of the LCDP.

ABSTRACT

Objective: Development and evaluation of a Descriptive Cognitive Model (DCM) for the identification of three types of usability issues in a low-code development platform (LCDP).

Background: LCDPs raise the level of abstraction of software development by freeing end-users from implementation details. An effective LCDP requires an understanding of how its users conceptualize programming. It is necessary to identify the gap between the LCDP end-users' conceptualization of programming, and the actions required by the platform. It is also relevant to evaluate how the conceptualization of the programming tasks varies according to the end-users' skills.

Method: DCMs are widely used in the description and analysis of the interaction between users and systems. We propose a DCM which we called PRECOG that combines task-decomposition methods with knowledge-based descriptions and criticality analysis. This DCM was validated using empirical techniques to provide the best insight regarding the users' interaction performance. Twenty programmers (10 experts, 10 novices) were observed using a LCDP and their interactions were analyzed according to our DCM.

Results: The DCM correctly identified several problems felt by first-time platform users. The patterns of issues observed were qualitatively different between groups. Experts mainly faced interaction related problems, while novices faced problems attributable to a lack of programming skills.

Conclusion: Applying the proposed DCM we were able to predict three types of interaction problems felt by first time users of the LCDP.

Application: The method is applicable when it is relevant to identify possible interaction problems, resulting from the users' background knowledge being insufficient to guarantee a successful completion of the task at hand.

Keywords: End-User Development, Low-Code Development Platforms, Descriptive Cognitive Models, Usability, Human-Computer Interaction

INTRODUCTION

Low-code development platforms (LCDP) address the need for increased productivity in software development. By raising the abstraction level at which software is developed, they automate low-level and routine development tasks, effectively contributing to solve the problem of global shortage of professional software developers. Forrester's Low-Code Market Forecast predicts low-code platforms will reach over 15 billion US dollars in 2020 (Marvin, 2018). At the same time, they lower the entry barrier to software development. As these low-level tasks become automated, developers are not required to carry them out (or even know how to carry them out). Low-level technical details are effectively hidden by the platform. If the entry level becomes low enough, we can say these platforms become End-User Development (EUD) platforms (Fischer, Giaccardi, Ye, Sutcliffe, & Mehandjiev, 2004). At that point, no special programming skills are needed to use them. Other terms have been used to describe related concepts with varying levels of scope, such as End-User Programming (EUP), End-User Software Engineering (EUSE) and Meta-Design (see Barricelli, Cassano, Fogli, & Piccinno (2019) for a recent systematic review of the literature).

Whether considering LCDP or EUD, the users' prior knowledge plays a relevant role in the learning and using of a platform, as it will affect the way users approach the platform (Dijkstra, 1982). In the case of LCDP, there is the double challenge of supporting users with little or no knowledge of programming, while also supporting expert programmers. Indeed, understanding individual differences and expectations, and identifying the sources of variation among different users will help this type of platforms to be more broadly adopted (Blackwell, 2017). Since low-code development platforms aim at reducing the learning burden while providing powerful tools to address a wide range of problems, a trade-off must be established between the scope of application and the learning costs of the platforms and their languages. This necessarily implies building an understanding of how different types of users approach the platforms.

Descriptive Cognitive Models (DCM) can be used to study the interaction between one interactive system and its users, in particular to analyze how the interplay between the users' cognitive processes and the user interfaces' design might lead to faulty interactions or use errors (Nielsen, 1994). Its applicability to reason about the act of programming has long been explored (cf. Blackwell, Petre & Church, 2019). Nevertheless, in spite of relevant Human-Computer Interaction (HCI) findings and developments since the 1980s and recent developments in both LCDP and EUP, there is still a considerable number of relevant gaps in current knowledge about how people reason during programming and development tasks (Sajaniemi, 2008). According to Myers, Pane, and Ko (2004), conventional programming languages require the user or programmer to make "*tremendous transformations*" (pp.48) from what he or she intends to accomplish, to what he or she should code. Visual modelling languages, typically adopted by low-code development platforms, aim to mitigate this problem, but their actual effectiveness is still subject to debate.

The distance between the mental and the physical spaces in software development was the motivation behind the current work. More specifically, the long-term goal of this work is to support lowering the learning curve of a specific LCDP to the point that non-programmers (i.e., end-users) might use it to develop software (in practice, turning it into an EUD platform). The challenge then, is how to reduce the learning effort of users without reducing the scope of the possible application domains. As a contribution to this long-term goal, the work described in this paper aimed at understanding the difficulties faced by potential programmers with different expectations and academic backgrounds when using a specific LCDP. To achieve this, we developed a new descriptive cognitive model with the purpose of predicting usability issues in a LCDP.

THE LCDP – Low-Code Development Platform

A low-code development platform supports the development of software applications resorting to minimal code writing. Its objective is to empower different kinds of users, by allowing them to easily and quickly create applications: experienced users (e.g., programmers) are able to create software by writing considerably less code, while users without prior experience will require less formal training to start creating applications.

Due to non-disclosure agreement conditions, we are not authorized to name the LCDP under study, and for that reason it will henceforth be referred to simply as the LCDP. The LCDP under study allows developers to create both full stack web applications, and mobile applications. It provides a set of predefined templates to bootstrap the development process, which creates the base application. Developers can then expand the application on top of that. The development process itself is performed by resorting to high level development languages, mainly visual languages, similar to Unified Modeling Language (UML) diagrams (Fowler, M., & Kobryn, 2004). The platform also allows developers to graphically edit the interfaces and automatically generate pages and components (e.g., through drag and drop interactions). With this LCDP, it is possible to develop enterprise-grade level applications thanks to the integration mechanisms provided, for instance, with web services, databases or external systems (e.g., SAP).

Different languages with different abstraction levels are provided to define different components of the system. The definition of some aspects of the system, such as navigation between screens, the behavior of the screens and buttons, is done through a statechart-like language, as they are adequate for control-flow modeling. These diagrams have a simple syntax, which has the objective of being easily understood by a large audience. Some more complex aspects, such as data retrieval from a database, resort to a Domain Specific Language (DSL), which is more powerful, but simultaneously more complex. The platform also takes advantage of widely known formats, such as spreadsheets, in order to speed up the development process. This empowers those users who are non-experienced developers, but have had previous contact with these technologies, to more easily understand the platform. Once finished, the applications are converted into standard technologies (familiar web, back-end and mobile languages), and deployed into a cloud environment. The applications become immediately available once published. Examples of this type of platforms include Appian, Google App Maker, Microsoft PowerApps, MIT App Inventor, Nintex Workflow Cloud, OutSystems, Sysdev Kalipso or Zoho Creator.

Descriptive Cognitive Models

At the dawn of HCI as an independent discipline, Richard Young wrote that "*for an interactive device to be satisfactory, its intended users must be able to form a 'conceptual model' of the device which can guide their actions and help them interpret its behavior*" (Young, 1981). Since then, it is commonly agreed that knowledge about how users perceive and interact with a computerized environment is of the foremost importance in the design of computer systems that emphasize usefulness and usability (Silva, 2013). The development process of an interactive system greatly benefits from putting the human, the user, in a central position during discussion and design (Dix, Finlay, Abowd, & Beale, 2004; ISO, 2010). In order to better understand how the user conceptualizes and interacts with a system, the discipline of HCI often resorts to models.

Descriptive Cognitive Models (DCMs) are widely used in the study and development of interfaces. Their analytical processes have since long been applied by experts, analysts and developers in order to obtain insight on how the interaction flow, the design features, or the information content of an interface might lead to performance deficits, faulty interactions or use errors (Nielsen, 1994). Although a comprehensive set of DCMs have been developed since the 1980s, it is usually a combination of different models tailored to a specific application case that provides the best result (i.e., insights on how users are thinking about the system and the interaction process). For the purpose of predicting usability issues in LCDP we have selected task decomposition models for a recursive decomposition of our main task into sub-tasks; knowledge-based analysis to comprehend the user's knowledge about the objects and actions involved in a given task; and risk assessment to analyze and evaluate the risk associated with the identified issues.

Task Decomposition

The process of describing the interaction process is often referred to as *Task Analysis* and consists of detailed descriptions and analysis of how people perform their jobs or tasks. It details what they do, what they act on and what they need to know. Identifying the elements and the goals of the task is an essential step to examine the skills necessary to perform a given job. Task decomposition can be performed either in the design phase of a new system or to suggest changes in an existing system.

Hierarchical Task Analysis (HTA)

The purpose of HTA is to decompose a task into all its sub-tasks in a way that displays the hierarchical relation between them. It is one of the most predominant examples of a task decomposition methodology. The outputs of HTA are a hierarchy of tasks and sub-tasks, together with plans describing in what order and under what conditions sub-tasks are performed. For examples and further details please see Dix *et al.* (2004) and, for a review on different ways of presenting an HTA and a proposal on an updated notation, see Huddlestone and Stanton (2016).

Knowledge-based analysis

The aim of a Knowledge-based approach to task analysis "*is to understand the knowledge needed to perform a task*" (Dix et al., 2004). The main goal of this type of analysis is to build general knowledge taxonomies for each task, after listing all objects and actions. Programming is a knowledge-based activity, and for the purpose of this study we will focus on analyses designed to predict difficulties from interface specifications, namely the External-Internal Task Mapping Analysis.

External-Internal Task Mapping Analysis (ETIT)

The ETIT model attempts to deal with the mismatch between the way the user thinks, and the way a system is designed, stating that this mismatch continues until the user learns how to translate what he or she wants to do into the system's terms. ETIT is a contribution of one of the most seminal authors in HCI, Thomas P. Moran (Moran, 1983). In his paper, Moran points out the need for the users to map between the task they are performing and their conceptual model of the machine. Thus, ETIT was conceptualized as a way of assessing the (1) complexity of learning of a naïve user or (2) transfer of knowledge between different systems. In the first case, which is the focus of the current work, ETIT assumes two different spaces: 1) the external task space (i.e., the naïve user's mental model of the task) and 2) the internal task space (the system's commands that allow it to perform the task). The relation between both spaces will be an indicator of the difficulty found in learning how to use the system. According to Moran (1983), when people start using a system, they know they must convert the tasks they have to perform into the system's language and concepts, i.e., they must learn to translate what they want to do into the system's terms. In this model, this translation is represented by *mapping rules*.

The ETIT analysis has three parts:

1. *An external task space (concepts and tasks described in those concepts)* – whenever a person needs to perform a task, this task is formulated in the external domain/real world, not in the system's terms (Moran, 1983). This means that people formulate their own mental model of the task, using their own known concepts, words and logic;

2. *An internal task space (concepts and tasks described in those concepts)* – the system's commands and interaction flow that allows the user to perform the task;

3. *A mapping* from the external task space to the internal task space.

While the external space is rich and diverse, systems are not. Systems usually abstract a small set of primitive concepts, converting the external task space into smaller internal task spaces. Particularly relevant here is that, while ETIT was conceived as a tool for system design, it can also be used as a competence model of the user, because it makes it explicit what is the knowledge necessary to execute a task.

Risk Assessment

The term "risk assessment" is usually connected to occupational health and safety but in this case, the methodology will be applied to the identification of usability and interaction issues that have the potential to compromise the application under development. The aim of such a process is firstly to identify the issue, and then to mitigate its consequences by adding control measures (Amir-Heidari & Ebrahemzadih, 2015). The steps to be applied in our descriptive cognitive model consist of:

- *hazard identification* - finding, listing and characterizing issues
- *risk analysis* - determining the likelihood of the issue
- *risk evaluation* - comparing an estimated issue against predetermined risk criteria to determine the significance or criticality of the issue

After performing a risk assessment, a remedy analysis can be performed where error reduction strategies are defined.

Proposed model - PRECOG: low-code development Platform descRiptivE COGnitive model

The descriptive cognitive model we propose was named PRECOG – low-code development Platform descRiptivE COGnitive model (Figure 1). From an analysis of the summarized methods, it became clear that these might interplay to account for an informative Descriptive Cognitive Model. In order to apply the proposed model, the analyst should start by performing an HTA of the particular development use case under analysis. The output of the HTA will provide a list of sub-tasks of the use case that need to be further analyzed. This will be done according to an adapted version of the ETIT. In this adapted version, sub-tasks will be described from the perspective of a naïve user's mental model, using its own terms and concepts (External task space, henceforth the Knowledge-Based Description - KBD), and from the perspective of an internal task space (henceforth System-Based Description - SBD) which details the steps needed in the LCDP, in order to accomplish the sub-task.

The KBD relies on user's data prior to any interaction with the system. With only a brief description of each task, participants should describe how they would reach each task's final goal using their current knowledge and familiar development tools. This information provides the analysts with the participant's mental model of the tasks in hand, before interaction with the system under evaluation.

The SBD, on the other hand, is a step-by-step description of the actions performed by an expert user of the system. For the purpose of comparability, the evaluator should define the same decomposition stop-condition for both descriptions. One fitting criterion for a LCDP HTA stop-condition is a discernable user interaction capable of being recorded by the platform (e.g., a drag-and-drop action; the selection of an item from a drop-down menu; the establishment of a new connection in a state-chart; the writing of an expression to define a condition). User interactions at a more atomic level, such as mouse movement, hovering, or typing of a specific character, do not need to be specified in HTAs for LCDP.

Mapping rules should be established between the naive user's mental model (KBD) and the system-based description (SBD), in order to identify possible conflicts. According to our predictions, three types of conflicts might be uncovered by looking at the mapping between the two spaces:

1. ***Under decomposition conflict*** - occurs when a procedure that is considered by the user as a single step in the knowledge-based description requires multiple steps in the system-based description. This type of conflicts might lead to an underestimation of the sub-task's complexity.

2. ***Over decomposition conflict*** - occurs when a procedure that is considered by the user as having multiple steps in the knowledge-based description requires only one step in the system-based description. This type of conflicts might lead to an overestimation of the sub-task's complexity and failure to identify and take shortcuts during the development task.

3. ***No Correspondence conflict*** - occurs when there is no link between a step in the knowledge-based description and one (or several) steps in the system-based description. This might occur because the user is not aware of the appropriate steps required by the LCDP or because the system does not include a feature representative of a mental step the user thinks is needed.

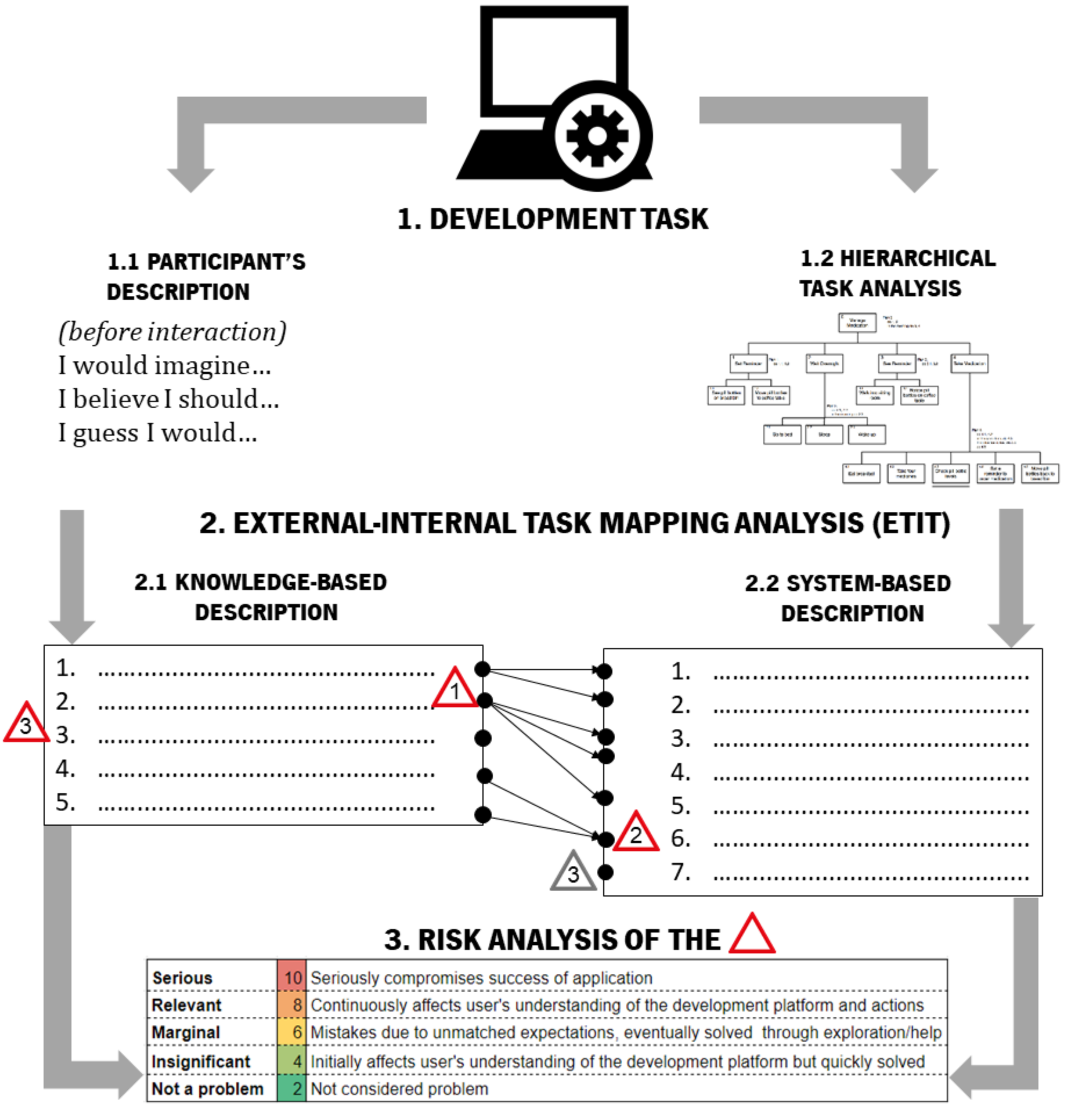

Figure 1 *– The proposed methodology applied to a given subtask of the HTA. Red triangles signal identified issues that occurred during interaction and the numbers inside correspond to the type of error (1 - Under decomposition; 2 - Over decomposition; 3 - No correspondence). Triangles are colored grey if, after empirical tests with a user, the analyst finds that the predicted error did not occur (false positive).*

Once one of these types of conflicts is found, a risk analysis of the conflict should be made. This risk analysis is usually performed in safety critical scenarios, which is not the context of the current study. Nevertheless, for the purpose of providing a richer PRECOG analysis, the risk analysis step can help define priorities in the continuous improvement of the system.

Figure 1 summarizes the steps of the overall analysis process. First, a development task is selected (in this case, *create a user interface to list and search for books*) and two branches originate from this task. On the top-right hand side of the figure (Step 1.2), an example HTA is presented for the development task. The HTA will generate the System-Based Description (SBD) or Internal Task Space for the selected task. On the left (Step 1.1), the Knowledge-Based Description (KBD) or External Task Space consists in a description made by people who have not interacted with the low-code system, listing how they would expect to perform the task at hand, knowing what they know at the moment. The second step (Step 2 in the figure) consists in comparing the KBD with the SBD per participant and identifying conflicts that might occur in the interaction using the conflict taxonomy described above ((1) *under decomposition*, (2) *over decomposition*, and (3) *no correspondence*). In this example, four issues arise and are represented by the four triangles. The triangles are red if that issue eventually occurred during interaction, and are marked as grey if they did not occur, being classified as a false positive. *Under decomposition* and *over decomposition* markers are placed inside the box, and *no correspondence* markers are placed on the leftmost border of the boxes.

The third and last step in Figure 1 consists of a risk analysis of the identified issues. The risk analysis of a conflict includes:

- **Frequency** - An ordinal scale from 0 (never) to 5 (frequent)
- **Criticality** - All issues are evaluated regarding the level of criticality as described in Figure 2, ranging from 2 (not a problem) to 10 (serious) (Figure 2).
- **Pure Risk** - The Pure Risk value of the error is a single value representing the weight that should be attributed to an error, and it results from the intersection between a value of Frequency with a value of Criticality from the matrix of Frequency with Criticality adapted from Amir-Heidari and

Ebrahemzadih (2015) (Figure 3). The value ranges from 0 to 50, and the higher the value, the more Critical should the issue be considered, with a higher priority for intervention.

Serious	10	Seriously compromises the success of the application
Relevant	8	Continuously affects the user's understanding of the development platform and actions
Marginal	6	Mistakes due to unmatched expectations, eventually solved through exploration/help
Insignificant	4	Initially affects the user's understanding of the development platform but quickly solved
Not a problem	2	Not considered a problem

Figure 2. *Descriptors for each level of Criticality, adapted to the analyzed use-cases*

					Frequency (N)
50	40	30	20	10	5 Frequent 8 to 10 occurrences
40	32	24	16	8	4 Probable 7 to 8 occurrences
30	24	18	12	6	3 Occasional 5 to 6 occurrences
20	16	12	8	4	2 Remote 3 to 4 occurrences
10	8	6	4	2	1 Rare 1 to 2 occurrences
0	0	0	0	0	0 Never
10 Serious	8 Relevant	6 Marginal	4 Insignificant	2 No problem	
Criticality					

Figure 3. *Matrix of Frequency with Criticality adapted from Amir-Heidari and Ebrahemzadih (2015). The intersection between a value of Frequency with a value of Criticality provides the Pure Risk of the issue, a single value representing the weight that should be attributed to the issue. The higher the value, the more critical should the issue be considered.*

A model of this sort should be developed for each sub-task deemed relevant for analysis. This will provide relevant information about the participants' difficulties in mapping their conceptual model of the task to the system's operational environment.

Applications of the PRECOG model

PRECOG can have two main applications. It can (1) be used retrospectively to understand the root-cause of an issue identified through user testing, or (2) it can be used predictively to understand which potential issues are going to arise.

In the first case, the model is used to analyze and understand the problems identified during user testing. These problems can be tracked by identifying the correspondence conflicts in the previously made ETIT mapping. This helps to identify potential causes for the problems in terms of mismatches between the KBD and the SBD. The number of observations of the different errors provides additional information that is then used in the risk analysis (as the frequency of occurrence of the errors).

The second application for PRECOG is to use it as a predictive model, which can provide valuable information without the time-consuming process of data gathering with real users. This application takes advantage of the fact that the mapping of the participant's (knowledge-based) description to the platform's requirements provides a first prediction of the effect of the differences between the user's knowledge and system's requirements to achieve a given task. In this case, instead of evaluating the criticality of the issues that effectively occurred, an expert analyst walks through the LCDP and decides on the likelihood of the identified potential problems, evaluating their probability of occurrence given the platform's design. This evaluation should consider, for instance, visual aids and widgets that are available on the platform, and which might be helpful in solving the issue under analysis. This stage of the process refines the predictive power of the model, as it eliminates false positives identified in the mapping stage. Calculating the Pure Risk of all identified errors is then done resorting to an adapted version of the matrix in Figure 4, using Probability (Never, Low, Medium, High) instead of Frequency. The end result will be a list of potential errors organized by Pure Risk evaluation. Remedy analysis of relevant issues (for instance, all Marginal, Relevant and Serious issues would be further detailed) might then be performed, resorting to either expert evaluation or empirical analysis.

Application and Validation of PRECOG in Empirical user studies

In this section, we present an application of the PRECOG model where we will detail the analytical process of constructing the HTA, the adapted ETIT analysis to map user knowledge and system-based descriptions, and finally the risk analysis of the identified potential interaction problems. The presented results correspond to the validation of the PRECOG model through empirical user studies. They allow us to understand the suitability and viability of using the selected techniques for the analysis of interaction conflicts in a low-code development platform, including the impact of the identified problems and the analysis of their root-causes.

METHOD

The first phase of the application of the PRECOG model consisted in defining, with a professional user and LCDP platform developer, a set of representative tasks which could be performed by different types of users. After all tasks were defined, this expert user performed them, and the performance was later analyzed in detail in order to obtain a HTA of each task. The HTA also provided the basis to develop the System-Based Description to be used in the model for each task.

The second phase of the application was performed after empirical user studies with 20 participants. Besides providing usability metrics of performance, these user studies allowed the authors to gather the Knowledge-Based Description of each participant, collected prior to any contact with the LCDP. The user studies complied with the American Psychological Association Code of Ethics, and an informed consent was obtained from each participant.

With both the Knowledge and System-based descriptions, it was possible to do the mapping between both, applying the PRECOG to each participant, and listing all the potential issues and mistakes that could happen during task execution.

The final phase of the validation effort was carried out after thorough video analysis of each participant's performance and comparison between the model's prediction and the real outcome in the user studies. Having identified all observed issues, risk analysis was applied to understand the issues' frequency and criticality.

Participants

A total of 20 participants were recruited (Table 1). The recruited population respected the following requirements:

- 10 participants had a software-engineering background (formal education in the past or present) - denoted as **Experts**;
- 10 participants had education in social sciences, economics or finance areas - denoted as **Novices;**
- All participants were over 18 years old, proficient in English, unfamiliar with the LCDP (had never worked with it), and willing to accept the sessions to be recorded (screen and audio).

The recruitment was made via internal mailing lists at the author's institutions as well as personal contacts. Of the 20 participants, 7 were female and 13 were male.

Table 1 - *Characterization of the participants*

		Novices	*Experts*
Gender	Female	6	1
	Male	4	9
Degree		Psychology, Economics, Biochemistry, Management, Acoustics	Software Engineering

The recruitment phase included a questionnaire to understand the participants' experience with programming languages. This questionnaire was custom-made, inspired by the knowledge acquired by students during an informatics degree, and was divided into three sections, each with increasing complexity in terms of computer science skills. The first section tested if the participant was familiarized with EUD tools (specifically, Spreadsheets editing software), and basic computational concepts, such as the concept of formula. The second section tested the capability of the user to understand simple software development concepts, such as interpreting and writing software, elementary data structures (e.g. arrays and binary trees), and query languages. The third level tested if the user had advanced software development skills, such as communication protocols, object oriented concepts and software modelling. Participants should also indicate which programming languages and integrated development environment (IDE), if any, they felt comfortable using.

Defining the System-based description

The participants' use-case consisted in creating a web application to manage books. The development of the application ("My Books") was divided into five tasks, which could be performed in any order, each aimed at fulfilling one of the following requirements (in order of increasing difficulty):

1. The user of the "My Books" application should be able to list and search all books;
2. The user should be able to see and edit the details of a book;
3. The user should be able to register new books in the application;
4. The application should present the user with a homepage with two buttons:
    a. One that redirects users to the list of books;
    b. Another that goes to the screen that allows registering a new book;
5. When seeing the details of a book, the user should see a list of other books from the same author.

We started by developing an HTA for the overall use case, divided into 5 HTA sub-diagrams corresponding to the five different tasks of the use case. Figure 4 presents the HTA diagram for the first task, following the graphical notation of Marshall *et al.* (2003), where the main task description is at the root of the diagram ("0. List and Search all books") and the different sub-tasks at lower levels (e.g., "1. Create book representations").

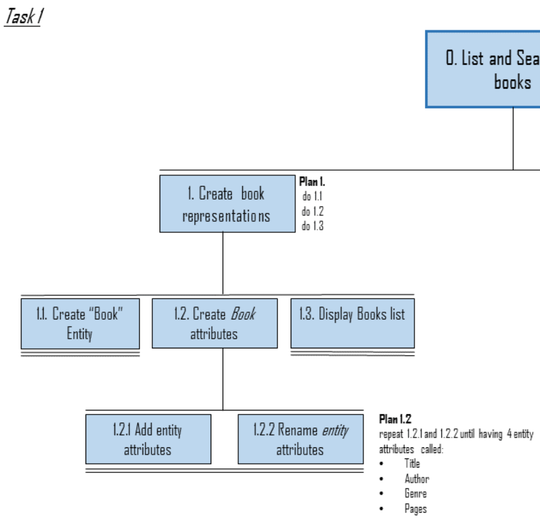

Figure 4 - *Example of one HTA diagram for the use case task "List and Search all books". When a (sub)task is further decomposed, a plan describing how its subtasks can be combined is detailed*

After defining the HTA for each sub-task, we defined the level of the HTA tree that better represents distinguishable tasks in the interface and we list all the units of interaction that are required to perform in the LCDP in order to complete that specific sub-task. For instance, to create a list of books with a search function the user of this particular LCDP as to perform the following actions:

1. Go to Data Menu;
2. Select Database;
3. Add Entity named "Book";
4. Add Attributes;
5. Rename Attributes;
6. Add Book Entity to Home Screen representation;
7. Boilerplate generation of search functionality

Defining the Knowledge-based Description

In order to obtain a Knowledge-based Description, each participant was instructed to verbally describe how he or she would perform each task, both in terms of the interface and in terms of the back-end development. Participants did this with the conceptual knowledge they had and using the tools they knew (if any), they should describe *how they would* complete the tasks. This was requested before the participant interacted with the LCDP.

Material

The tests were performed in quiet testing rooms. Locations were equipped with a table and three chairs, a laptop computer for the participant, with the LCDP running, ActivePresenter 7 screen and audio capture software (Atomi Systems, 2019), and a video camera (the participant's facial expressions were not captured at any time, the camera focused on the screen of the laptop as a backup measure).

Procedure

The participant was welcomed by the test moderator and the data logger, who explained that an evaluation was being carried out on how hard or how easy it was to use a particular LCDP. He or she read and signed the informed consent where more detailed information was provided. Each participant was presented with the five tasks. First, the participant was instructed to describe verbally how he or she would perform each task, both in terms of the interface and in terms of back-end development. Regarding this process, it became evident that the test moderator played a role in the success of this phase, which would be used to build the Knowledge-based Description of each participant. The moderator should prompt the participant when he or she becomes quiet, asking specifically about aspects of the application in order to gather as much information as possible.

Then, the participant was requested to carry out the tasks upon performing a tutorial. The tutorial consisted of an interactive session, where the users were put in contact with the LCDP and explained the basic concepts. The test moderator informed the participants that there were several ways of concluding the tasks, and that they could search the internet for answers. Each participant was provided with the same instructions. The participants were given a written copy of the instructions and respective memory aides.

Analysis

For each subtask in the Hierarchical Task Analysis deemed relevant, we developed adapted ETIT mappings from the collected Knowledge-based Description to what was the required plan of action in the platform (the System-based Description). Figures 5 and 6 show as an example the ETIT mapping for the sub-task "List and Search all Books", for an expert and a novice user, respectively. The end-result of the mapping exercise allows the analyst to identify what type of use-errors might occur during that sub-task.

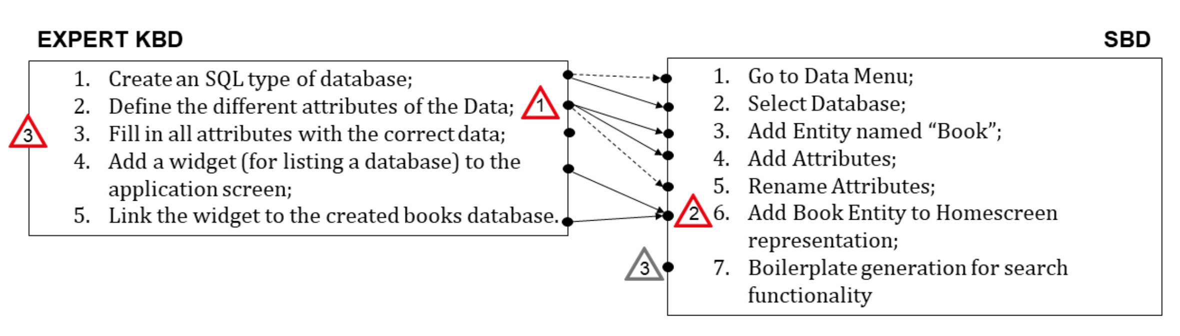

__Figure 5 -__ Example of the adapted ETIT mapping for the sub-task "List and Search all Books" in an Expert user. Dashed arrows correspond to implicit steps.

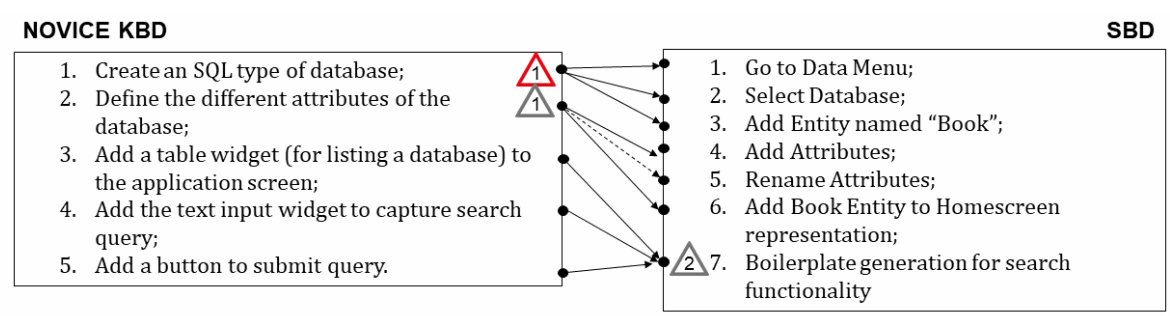

__Figure 6 -__ Example of the adapted ETIT mapping for the sub-task "List and Search all Books" in a Novice user.

As we pointed out earlier, analysts can uncover three types of conflicts by looking at the mapping rules. Figure 5 illustrates all three types of conflicts - *under decomposition* in step 2 of the KBD, *no correspondence* in step 3, and over *decomposition* in step 6 of the SBD.

Identify the root-cause of issues found in empirical tests with real users

The analysis depicted in Figure 1, which shows the complete flow from the definition of the models, through the identification of a KDB-SBD conflict, to risk analysis of that conflict, was the one followed in the present work.

Each interaction video of the participants was thoroughly observed, the issues that occurred were identified and their relevance analyzed using the following steps:

- **Root Cause** – After analysis of the interaction that resulted in an error, a root cause was identified.
- **Remedy Analysis** – A recommendation for a way to avoid the error or issue was devised.
- **Evidence** – When available, evidence gathered during data collection with participants (video) was also registered.
- **Pure Risk** – The frequency and criticality of the issues was evaluated by the authors and the LCDP's research and development team.

One complete analysis took on average 40 minutes per participant for an experienced analyst.

The Frequency of an issue was defined as the frequency with which the issue under analysis was observed in the empirical tests. The scale can be adjusted depending on the size of the sample, without the need to modify our model.

To determine Criticality, all observed issues were evaluated by four authors and four professional LCDP developers regarding the level of criticality as described in Figure 2, ranging from 2 (not a problem) to 10 (serious). The evaluations were performed individually considering the general criticality of the issue, and not the particular context/participant where it happened. Inter-rater reliability was assessed using a two-way, average measures ICC (intraclass correlation). The resulting ICC was in the "fair" range, ICC=0.50 (Cicchetti, 1994), indicating that evaluators had a fair degree of agreement. The ICC increased to 0.68, in the “good” range, when only one group (authors) was considered. This indicates different evaluation criteria from both groups, something which would be worth exploring in the future. The final Criticality value was the mode of the eight evaluations.

RESULTS

In this section we will present results on the comparison between the PRECOG’s predictions and the interaction issues observed during the empirical usability tests. Moreover, we will also address the nature of the use-errors that were predicted and verified, in terms of its root-cause, frequency, and pure-risk.

The comparison between PRECOG´s predictions and results of the empirical usability tests will be presented regarding the two profiles (expert and novice) and the three types of conflicts signalized by PRECOG (1 - Under decomposition; 2 - Over decomposition; 3 - No correspondence). In Table 2 we can see that out of a total of 135 potential interaction issues identified by PRECOG, 67 (49.6%) occurred during empirical usability tests. Moreover, of all interaction issues verified in the empirical usability tests, only 5 were not predicted by a type of conflict signalized in the PRECOG model. Table 2 summarizes the outcome of applying PRECOG in three confusion matrices, considering the studied profiles both combined and separately.

Table 2 - *Confusion Matrices for a combination of all participants and by participants’ profile (Novices and Experts).*

All Participants				
		Verified		
		Use Error	*No Use Error*	Total
Predicted	*Use Error*	67	68	135
	No Use Error	5	64	69
			TOTAL	204

EXPERTS				
		Verified		
		Use Error	*No Use Error*	Total
Predicted	*Use Error*	39	42	81
	No Use Error	2	50	52
			TOTAL	133

NOVICES				
		Verified		
		Use Error	*No Use Error*	Total
Predicted	*Use Error*	28	26	54
	No Use Error	3	14	17
			TOTAL	71

In order to assess the predictive capability of our DCM we analyzed the values presented in Table 2, where it is possible to see the total number of *true-positives* (i.e., predicted and confirmed use-error), *false-positives* (i.e., predicted but unconfirmed use-error), *true-negatives* (i.e., predicted and confirmed inexistence of use-error), and *false-negatives* (i.e., not predicted but confirmed existence of use-error). An efficient predictive model aims at scoring high in both true-positives and true-negatives and low in both false-positives and false-negatives. From a confusion matrix one can calculate several complementary values to assess a classifier's predictive capability (Powers, 2011; Tharwat, 2018), namely:

- *Sensitivity* - or *recall* is the proportion of the positive samples (i.e., verified use-errors) that were correctly classified as so. Thus, Sensitivity depends on true-positives (TP) and false-negatives (FN), which are in the same column of the confusion matrix, and can be calculated as: Sens = TP/(TP+FN)
- *Specificity* - or *inverse recall* is the proportion of negative samples (i.e., verified no use-error) that were correctly classified as so. Thus, Specificity depends on true-negatives (TN)

and false-positives (FP), which are in the same column of the confusion matrix, and can be calculated as: Spe = TN/(TN+FP)

- *Accuracy* - is defined as a ratio between the correctly classified samples to the total number of samples, and can be calculated as follows: Acc = (TP+TN)/(TP+TN+FP+FN)
- *F1-score* - also called F1-measure is the harmonic mean of sensitivity and positive predictive value (ppv = TP/(TP+FP)). The value of the F1-score ranges from zero to one, and high values indicate high classification performance. F1-scores are calculated as follow: F1 = (2TP)/(2TP+FP+FN)
- *Informedness* - also called *Youden's index* quantifies how informed a predictor is for the specified condition, and specifies the probability that a prediction is informed in relation to the condition (versus chance) (Powers, 2011). The value of the Informedness ranges from zero, or chance-level, to one, representing a perfect predictive capability. Informedness can be calculated as follow: Inf = *sensitivity* + *specificity* - 1

Table 3 shows the PRECOG's values obtained for these different variables, based on the confusion matrices presented in Table 2.

Table 3 - *PRECOG's predictive metrics for All Participants and divided by user profile*

	All Participants	*EXPERTS*	*NOVICES*
Sensitivity	0.93	0.95	0.90
Specificity	0.48	0.54	0.35
Accuracy	0.64	0.67	0.59
F_1 Score	0.65	0.64	0.66
Informedness	0.42	0.49	0.25

Both Powers (2011) and Tharwat (2018) discussed the advantages and limitations of each of these metrics for classification performance. According to Powers, Sensitivity and F1-scores ignore performance in correctly handling negative examples, propagate underlying marginal prevalence and biases, and fail to account for the change level performance. Nevertheless, Tharwat makes the case that all these different metrics, being more focused (such as Sensitivity, Specificity, and F1-score) or more general (such as Accuracy and Informedness) are useful to understand all the potentialities of a particular classifier. In the case of PRECOG, Sensitivity was generally higher than Specificity and, while Accuracy and F1-scores were fairly similar for both Experts and Novices, Informedness was the measure that varied the most regarding type of participant.

Statistical tests revealed differences between Experts and Novices concerning Specificity and Informedness. An unpaired two-samples Wilcoxon test indicated that Specificity in Experts was significantly higher than in Novices ($W = 73.5$, $p < .01$, $r = -0.59$). Similarly Informedness ($W = 81$, $p < .01$, $r = -0.72$) was also significantly higher in Experts than in Novices.

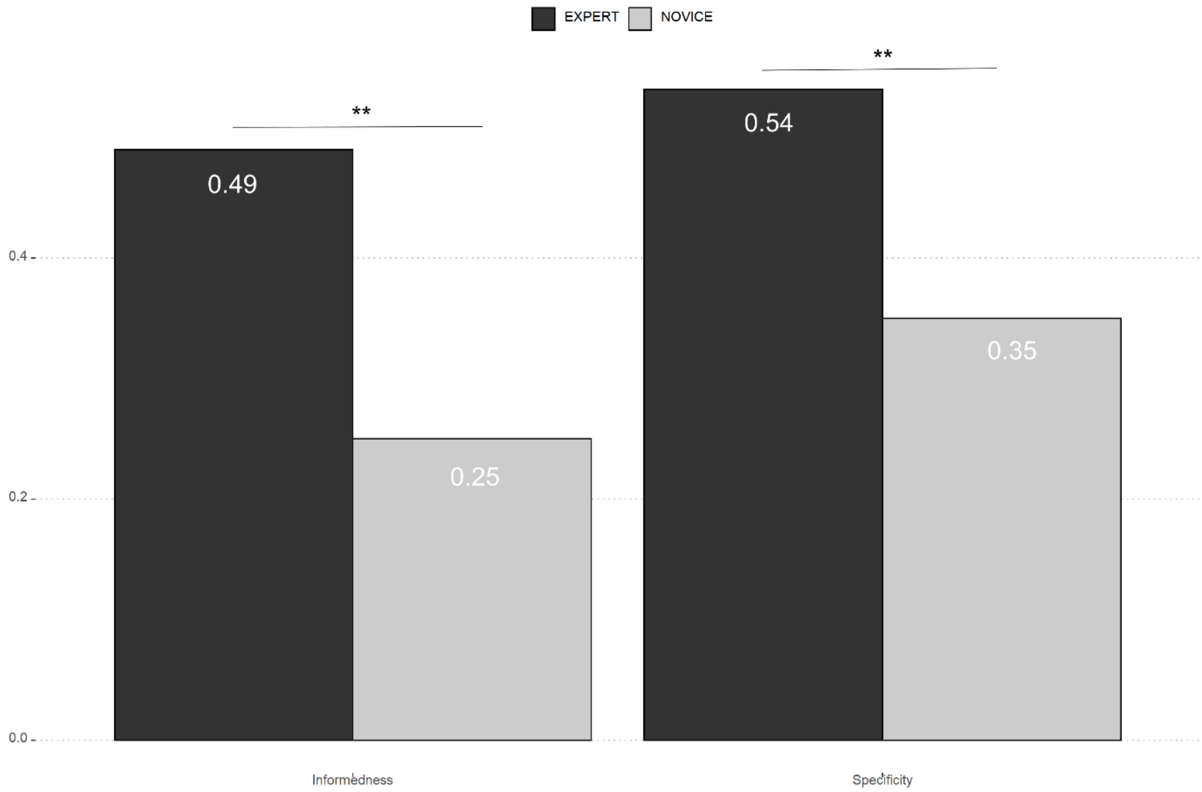

Figure 7 - *Differences PRECOG's classification of Experts and Novices concerning Specificity and Informedness.*

Having calculated the Sensitivity and Specificity of each participant's classification, it was possible to map the performance of PRECOG in a Receiver Operating Characteristic (ROC) plot (Figure 8). Participants are mainly mapped in the upper left-hand of the ROC space, meaning that, for the generality of the participants, PRECOG's classification was predictive of actual behavior.

Again, it is possible to verify that data points of Expert participants are further from chance level (x=y) than data points of Novice participants.

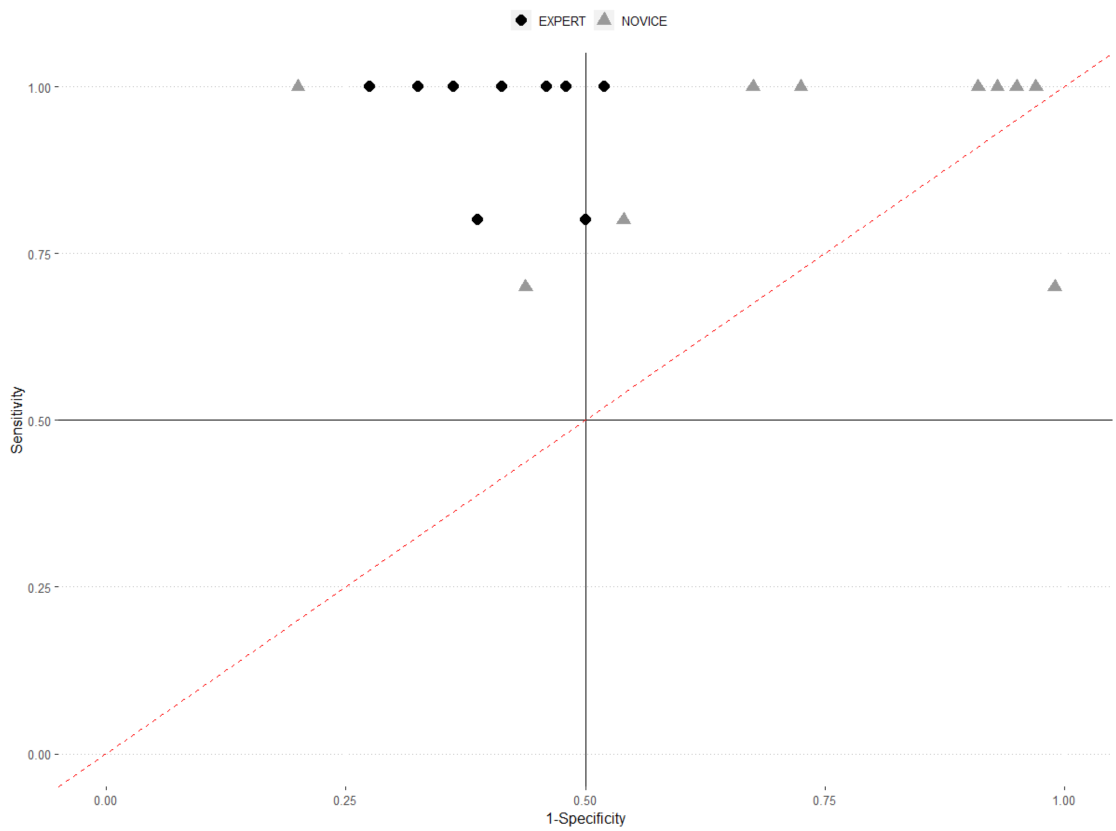

Figure 8 *- Receiver Operating Characteristic (ROC) data point cloud representing Experts (points) and Novices (triangles). The diagonal line represents the chance level.*

Looking at the distribution of True Positives and False Positives according to profile and considering the three types of conflicts predicted by PRECOG, it is possible to observe that Under decomposition conflicts were the type of mapping conflict where PRECOG performed better. PRECOG obtained the highest difference between True Positives and False Positives in this type of conflict, having Under decomposition conflicts in Experts accounting for 17,5% of the overall correct predictions and in Novices accounting for 23.6% of the overall correct predictions. In the case of Experts, Over decomposition got the highest rate of True Positives (20%), however, this type of conflict also accounted for 17.5% of False Positives. For the Over decomposition conflicts in Novices, the reverse pattern was observed, with a higher number of False Positives (18.1%) than of True

Positives (10%). Finally, the No correspondence type of conflicts were the type of mapping conflict where PRECOG had a worse performance, with a higher number of False Positives than True Positives both in Experts (20%) and in Novices (21%).

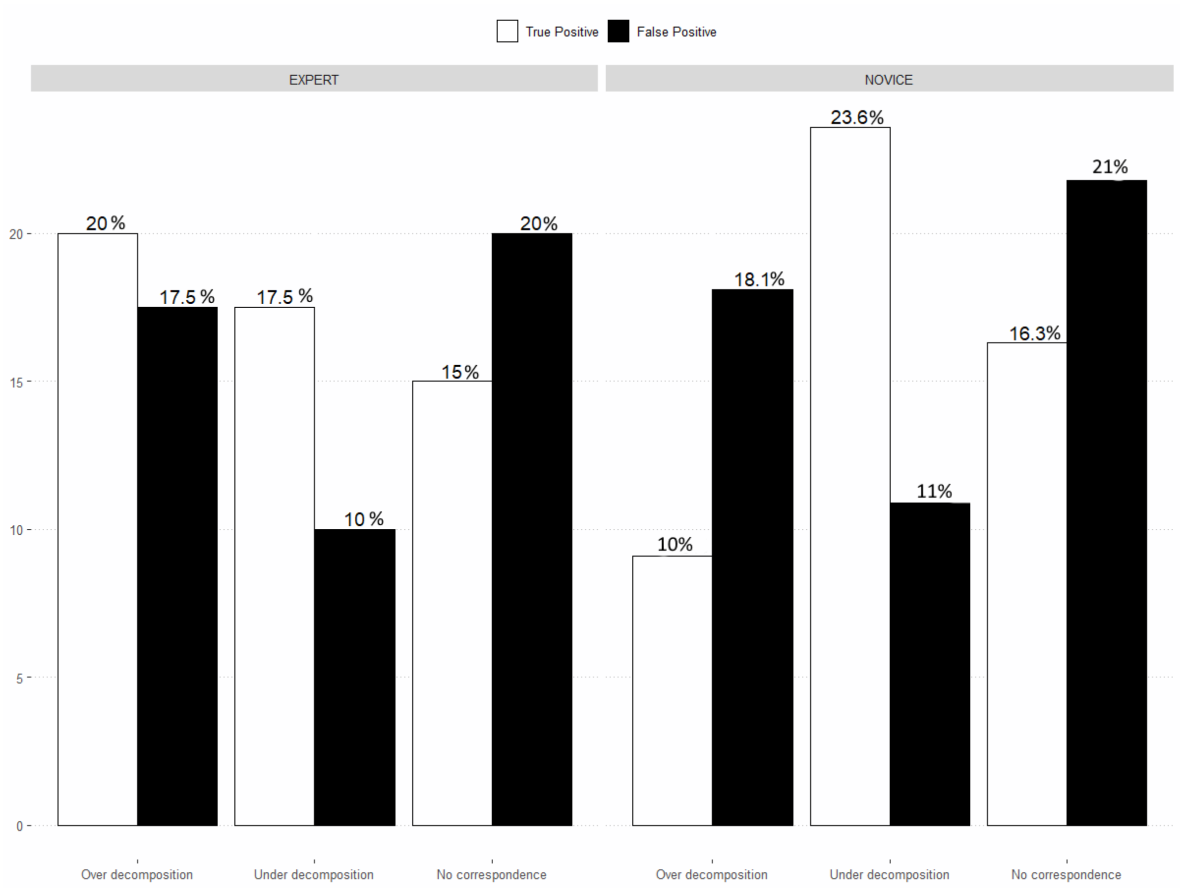

Figure 9 *- Distribution of True Positives and False Positives according to profile, considering the three types of conflicts predicted by PRECOG*

Focusing on the data coming from the usability empirical studies, and considering the Frequency of the observed interaction issues, we were able to identify 10 types of issues: Input parameter; New database; Table with data; Button link; New screen; Query; Search function; Relate data; Change homepage; Details. Figure 10 depicts the Frequency with which each of these issues occurred in both profiles. It is possible to observe that the same issues did not occur for both profiles. As a first characterization, it can be observed that Experts had a more diverse typology of issues, having experienced nine while Novices had six different types of issues. No participant had any "Button Link" related issues. On the Novice side, they did not have issues related to "Table with data", "Query" and "Relate data". For the Experts, the issue with the highest True Positives was "Query" with 8 correctly identified issues. The highest number of False Positives among Experts were found in the "New Screen" and "Change homepage" issues, where 7 identified issues were considered False Positives. Regarding Novices, "Input Parameter" and "New database" issues had 7 True Positives followed by "New Screen" with 6 True Positives. Concerning False Positives, "Search function" with 7 False positives was followed by "Details" with 6.

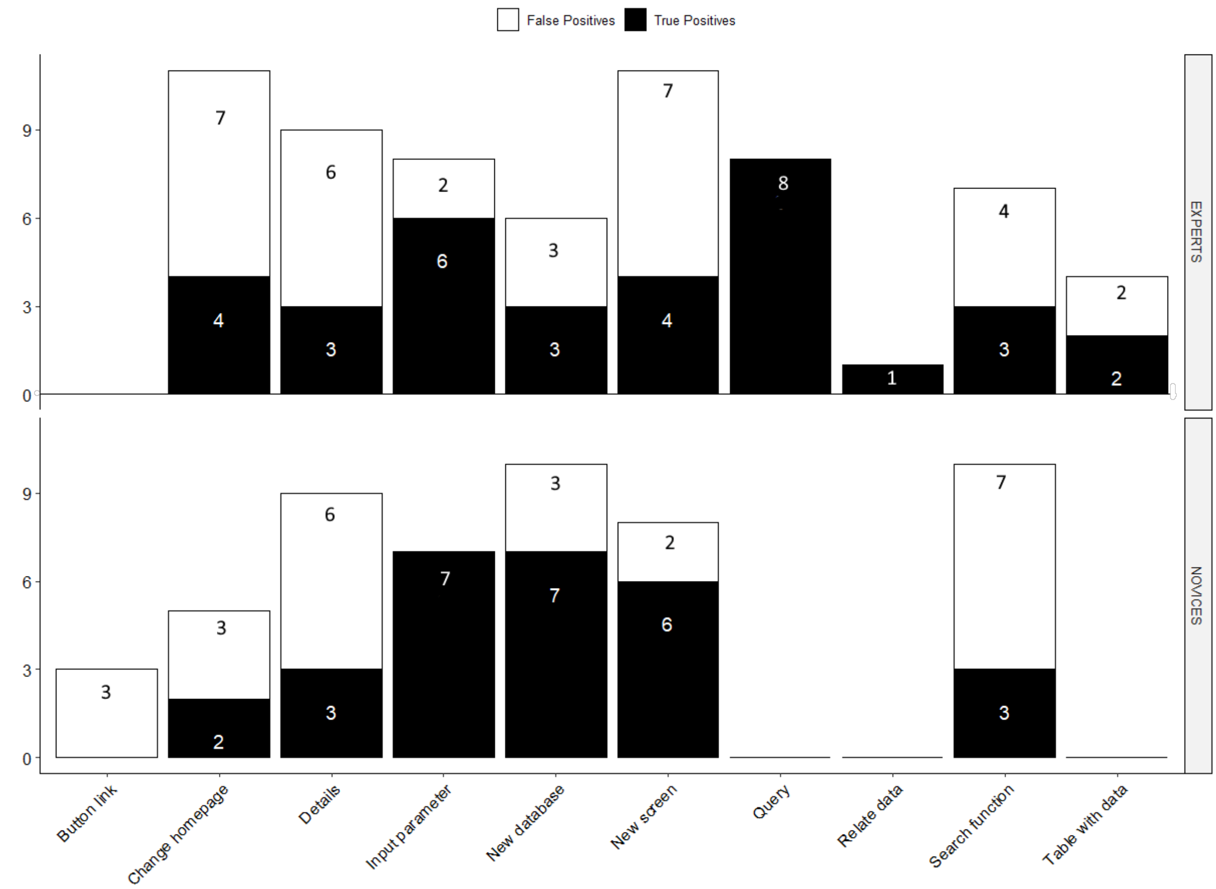

Figure 10 - *Frequency of True and False positive issues identified in the empirical tests after KBD-SBD mapping*

Besides Frequency, another component for the Pure Risk analysis is the Criticality of the issues. Table 4 presents the final Criticality attributed to the issues that emerged during our analysis.

Table 4 - *Criticality of the identified issues by a pool of eight evaluators*

Issue	*Criticality*	*Description*
Input parameter	8	Relevant
New database	6	Marginal
Table with data	8	Relevant
Button link	8	Relevant
New screen	8	Relevant
Query	8	Relevant
Search function	8	Relevant
Relate data	8	Relevant
Change homepage	6	Marginal
Details	8	Relevant

Figure 11 depicts all ten different issues observed during the interaction with the LCDP. The issues are presented in terms of the Pure Risk evaluation (min=0, max=50). Besides Criticality, the Pure Risk evaluation considers the Frequency of occurrence of each issue, hence the difference in the value of the same issue for both profiles.

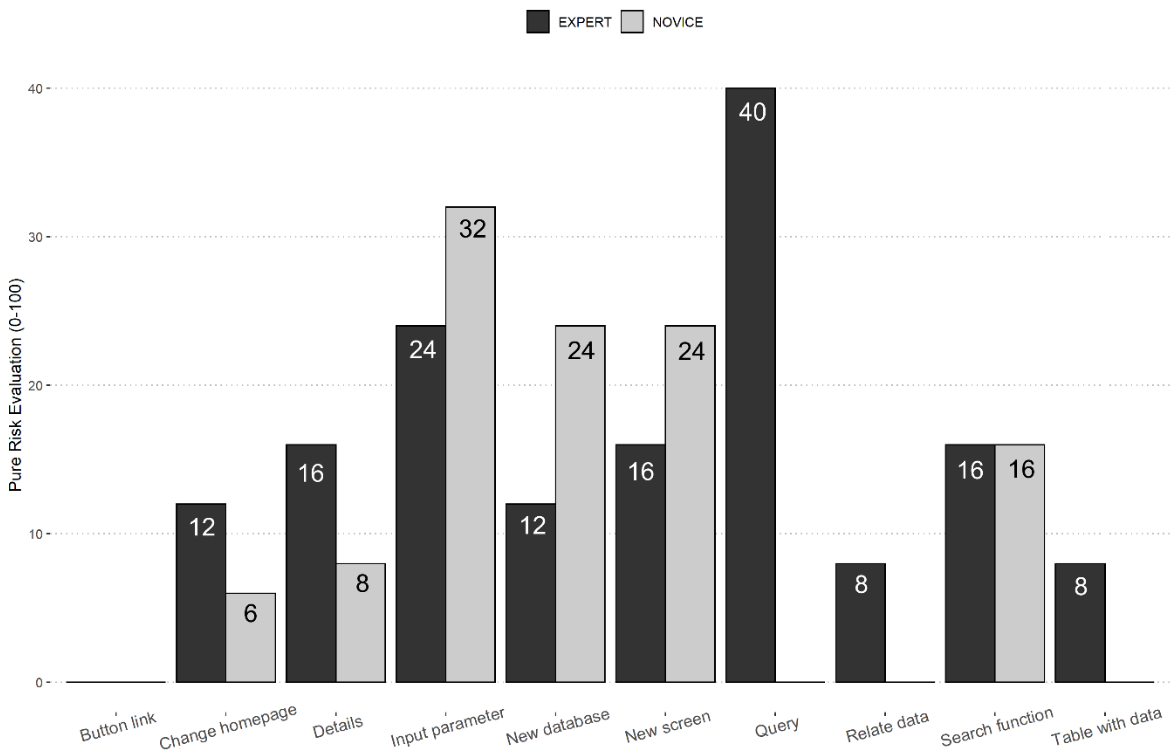

Figure 11 - *Pure Risk of each observed issue per profile. Issues found by the LCDP Descriptive Cognitive Model according to their Pure Risk evaluation (min=0, max= 50)*

This data indicates that, for instance, the “Query” issue should be solved, as almost all Experts who interacted with the LCDP had issues with creating a query. It also indicates that the weight of each issue is different depending on the LCDP user. For Novices, the highest Pure Risk value is in the “Input Parameter”, followed by “New Database” and “New Screen” issues. The only Pure Risk value similar to both profiles occurred in the “Search Function” issue, indicating similar difficulties in this task for both profiles.

DISCUSSION

We have successfully applied our descriptive cognitive model (PRECOG) to a relevant use case, which allowed us to validate the viability of the proposed approach.

Predictive power

Although time-consuming, the methodology proved to correctly identify several high-criticality issues from both user profiles. Indeed, we were able to use the model to predict a relevant number of problems, prior to the user studies that confirmed them. The confirmation came with a relatively high number of False Positives. Such is due to the conservative nature of the model, as highlighted by the fact that results show a high Sensitivity, with a comparatively lower Specificity. We opted for an exhaustive approach, and no relevant error was disregarded in the KBD-SBD mapping analysis, since such a stance allowed a more extensive list of expected problems. Despite this conservative approach, the majority of the predicted issues or root causes occurred during empirical user tests (all were observed except for the “Button Link” issue). This is a relevant output as very little unpredicted issues occurred during the user studies and these were mostly related to navigational (interaction with the platform) difficulties and not conceptual mismatches.

Another explanation for the high number of False Positives predicted by PRECOG concerns “No correspondence” issues, where one item from the KBD or the SBD finds no correspondent on the other side. Some issues of this type may have arisen because the participant did not mention a certain development step or minor detail - which ended up being evident when interacting with the LCDP, thus not resulting in a real issue. This is particularly evident in the case of Novice programmers, leading to a significantly lower Specificity in their case, and can be attributed to the fact that their mental models of the programming tasks differ from the platform's model more than the Experts' mental model. That some of the problems were not observed is, in effect, a positive indicator towards the ability of the LCDP to guide users during the learning stage. Conversely, "Over decomposition" was the type of issue with more false positives, which seems to indicate a need to raise the abstraction level of the platform. Doing this without compromising the ability of advanced users to fine-tune applications that are more complex, implies exploring strategies for adaptive user support (Oppermann, 1994; Gajos, Czerwinski, Tan & Weld, 2006). An important observation should be made regarding moderation of the Knowledge-Based Description stage. The moderator should be very familiar with the platform and the tasks under study in order to prompt the necessary information to complete the mapping. It is extremely important to obtain all the information that will allow illustrating the participant's mental model before interaction with the platform.

Another significant result concerns the Criticality of the issues observed. All issues which occurred had a Criticality evaluation of more than 6, that is, were either Marginal (*Mistakes due to unmatched expectations, eventually solved through exploration/help*), or Relevant (*Continuously affects user's understanding of the development platform and actions*). This is an indicator that our approach is useful in detecting high-criticality issues. Unfortunately, and this is a limitation of the current study, no Criticality evaluation was performed in the non-observed issues in order to compare the criticality between the observed and unobserved issues.

Regarding the number and type of issues, the applied model allowed us to distinguish two patterns between Novice and Expert programmers. Expert programmers had more types of issues (9 in total), but each happening less frequently, depicting a more exploratory behaviour, Novice programmers had less types of issues (6 in total) but each with more repetitions, meaning that the issues were effectively problematic for this profile, which explored less and whose participants had very similar performances. Another potential explanation for this difference in number of observed issues results from the speed with which each profile performed the tasks. Expert programmers managed to complete more tasks than Novice programmers, hence there was more opportunity for issues to arise. These different patterns affected the Pure Risk evaluation of the issues according to profiles. Even though the Criticality was the same for all issues independently of the profile, the Frequency of the issue affected the Pure Risk value. The issues found in Novice programmers were consistent and robust, and this is even more interesting if we consider their heterogeneous background education. The Pure Risk evaluation should work as an order of priorities in order to improve the LCDP under study. Indeed, the LCDP development team has since addressed some of the issues found.

In global terms, the combined results show PRECOG is somewhat a conservative model, which despite eliciting some False Positives also identified correctly the majority of issues that effectively impaired the participants' progression. It should be noted that we are considering first-time users of the platform without any formal training.

As it is the case for safety-critical systems, when analyzing applications such as LCDPs, it is safer to overestimate the occurrence of potential for use-error, than fail to identify serious use-errors that occurred. In this regard, PRECOG appears to be a promising approach in the sense that it was able to identify almost all issues faced by Expert and Novices programmers.

From the application of this LCDP Descriptive Cognitive Model, and as highlighted by the ROC analysis, we conclude that the method has predictive power (i.e., most of the identified knowledge-system conflicts were resulting in use issues during the performance of the task), and that this methodology could be used as an effective tool to predict, understand, and mitigate use errors and faulty interactions in an LCDP platform.

Model applicability

Regarding the applicability of PRECOG, the descriptive cognitive model itself was tailored to a specific set of applications, and for the LCDP it proved useful, granular and precise. In theory, the model is not limited to LCDP platforms. We address applications where the user tasks can be decomposed through an HTA, so that the KBD-SBD mapping is possible. That said, we are only able to support our claims in the low-code development area, due to the performed user studies. As for the type of participants, the model assumes naïve participants or first-time users, but maybe the observed granularity and detail of the predictions would allow its application with participants with some knowledge of the platform under study (i.e., testing tasks that the participants never performed in that particular platform). This would be an interesting application for the future, as we expect the model to identify the missing knowledge independently of the proficiency of the participant.

Threats to validity

Having performed a user study, results are susceptible to threats to its validity. Specifically, we acknowledge the relatively limited number of participants (a total of 20). While the achieved results regarding the percentage of effective issues vs. identified ones are positive, the impact of a larger study group should be analyzed.

Would this model work without empirical studies? We believe it would, and in the section where PRECOG is presented, we provide the necessary steps to do so. We believe its predictive capability would be improved if the criticality evaluation had been performed after the KBD-SBD mapping for all identified potential issues (including false positives). Having all the criticalities of all potential issues would, for instance, allow the analysts to choose the issues evaluated with a criticality of 6 or more. These issues, according to our results, have a higher probability of being captured by PRECOG. In the future, it is our intention to validate this second approach by performing cognitive walk-throughs in the platform and performing criticality evaluations in replacement of the empirical studies with real participants.

Value

The approach used in the current study allowed us to identify problems, difficulties and issues participants faced during the interaction with the LCDP. Although not detailed in the present study, a root-cause analysis of each issue allowed us to understand that these arose mostly due to two types of lacking concepts: LCDP-related concepts and development-related concepts. Whereas Expert programmers knew the development concepts but had difficulty in translating them into the LCDP terms, Novice programmers lacked basic development-related concepts, which largely affected their performance.

Although these results are based on the study of one development use-case, and we cannot generalize to the entire LCDP, PRECOG allowed the identification, prioritization and root-cause analysis of several issues. This is valuable information for the LCDP platform developers as they have as main goal to place both types of profiles (Experts and Novices) within the Optimal Flow (cf.

Csikszentmihalyi, 1990; Repenning & Ioannidou, 2006), with just the right amount of challenges and just the right amount of skill-acquisition - each at their own pace. The nature of the issues also provides valuable inputs to support adjusting the LCDP learning process, according to the Optimal Flow. Specifically, the nature of the errors should be taken into account: Novice users, due to the lack of software development skills, will fall into anxiety, as they are not able to develop the desired features; Expert users, while lacking knowledge of the platform, will perform the tasks resorting to the previously acquired knowledge, which might result in repetitive and monotonous tasks, leading to boredom.

CONCLUSION

Low-code development platforms have the potential to dramatically change how software is developed, making it possible, at least for particular domains, for someone without a formal education in computer science to develop quality software, and for experienced developers to significantly speed up the development process. Understanding how programmers and non-programmers approach this type of platform, is key to support their design and evolution. By developing and applying PRECOG, a new Descriptive Cognitive Model (DCM), aimed at identifying interaction issues with the learning of low-code platforms, we were able to gain insights into potential problems with a specific low-code platform's use. The proposed DCM was validated, using empirical techniques. Twenty participants were observed interacting with the LCDP, of which 10 were expert programmers and the other 10 were novice programmers. All performed the same tasks and all interactions were analyzed according to the proposed model.

Although a high number of False Positives were identified after a first mapping between the user's mental model and the system's requirements, it is relevant to notice that all issues but one (Button link), which occurred during users' interaction with the LCDP, were predicted by this mapping. Expert programmers had a higher number of observed issues, although each occurring less frequently. This was due to expert programmers performing the tasks more quickly and with a more explorative behaviour, giving room for more issues to occur. On the other hand, Novice programmers faced fewer issues, although each occurred more frequently. These results allowed us to successfully identify high criticality use errors through the analysis of the users' mental model and, importantly, the results allowed us to identify the root causes of each issue. One of the future goals of the current research is to validate PRECOG as a predictive model without recurring to user studies.

PRECOG revealed itself quite valuable in the search for more usable LCDP and effective EUD platforms. As Maeda (2006) points out, "*observing what fails to make sense to the non-expert, and then following that trail successively to the very end of the knowledge chain is the critical path to success* [i.e., in developing simple and easy to learn systems]". Our proposed method allows the systematic and effective exploration of the conflict between users' knowledge and system requirements/challenges, thus providing important insights for system developers that aim at creating a broadly accessible development platform. Moreover, this method can be applied in other contexts where learnability might be an issue for it allows to identify possible sources of faulty interaction and sub-tasks where the users' background knowledge will be insufficient to guarantee a successful performance of the task at hand.

KEY POINTS

- An effective Low-Code Development Platform (LCDP) requires an understanding of the distance between the LCDP end-users' conceptualization of programming, and the actions required in the platform.
- We propose and evaluate a Descriptive Cognitive Model (DCM) for the identification of initial use issues in a low-code development platform.
- We propose three mapping rules for the identification of knowledge-system conflicts: over decomposition, under decomposition and no correspondence conflicts.
- Applying the proposed DCM we were able to predict the interaction problems felt by first time users of the LCDP.

REFERENCES

Amir-Heidari, P., Farahani, H., & Ebrahemzadih, M. (2015). Risk assessment of oil and gas well drilling activities in Iran – A case study: Human factors. *International Journal of Occupational Safety and Ergonomics*, *21*(3), 276–283.

Barricelli, B. R., Cassano, F., Fogli, D., & Piccinno, A. (2019). End-user development, end-user programming and end-user software engineering: A systematic mapping study. *Journal of Systems and Software*, *149*, 101–137.

Blackwell, A. F. (2017). End-user developers–What Are They Like?. *New Perspectives in End-User Development* (pp. 121-135). Cham, Springer International Publishing.

Blackwell, A. F., Petre, M. & Church, L. (2019). Fifty years of the psychology of programming. *International Journal of Human-Computer Studies, 131*, 52-63.

Brooks, F. P. (1987). No silver bullet: Essence and accidents of software engineering, *Computer*, 20(4), 10-19.

Ceruzzi, P. E. (2012). *Computing: a concise history*. Cambridge, Massachusetts: MIT Press.

Cicchetti, D. V. (1994). Guidelines, criteria, and rules of thumb for evaluating normed and standardized assessment instruments in psychology. *Psychological assessment*, *6*(4), 284-290.

Csikszentmihalyi, M. (1990). *Flow: The Psychology of Optimal Experience*. New York: Harper & Row.

Dijkstra, E. W. (1982). How do we tell truths that might hurt? In *Selected Writings on Computing: A personal Perspective* (pp. 129–131). New York: Springer New York.

Dix, A., Finlay, J., Abowd, G. D., & Beale, R. (2004). *Human-Computer Interaction*. Harlow, England: Pearson Education.

Embrey, D. E. (1986). SHERPA: A systematic human error reduction and prediction approach. Paper presented at the International Meeting on Advances in Nuclear Power Systems, Knoxville, TN.

Fischer, G., Giaccardi, E., Ye, Y., Sutcliffe, A. G., & Mehandjiev, N. (2004). Meta-design: A manifesto for End-user development. *Communications of the ACM*, *47*(9), 33–37.

Fowler, M., & Kobryn, C. (2004). *UML distilled: a brief guide to the standard object modeling language*. Addison-Wesley Professional.

Gajos, K. Z., Czerwinski, M., Tan, D. S., & Weld, D. S. (2006, May). Exploring the design space for adaptive graphical user interfaces. *Proceedings of the working conference on Advanced visual interfaces* (AVI '06) (pp. 201-208). New York: ACM.

Gonçalves, R. C., Maciel, R. H., Maia, L. M., Nascimento, A. P. T., & Canuto, K. M. (2014). Electric system control room operators: Cognitive task analysis and Human error. *Proceedings of the 5th International Conference on Applied Human Factors and Ergonomics (AHFE 2014)* (pp.621-630).

Huddlestone, J. A., & Stanton, N. A. (2016). New graphical and text-based notations for representing task decomposition hierarchies: Towards improving the usability of an Ergonomics method. *Theoretical Issues in Ergonomics Science*, *17*(5–6), 588–606.

ISO - International Organization for Standardization. (2010). Ergonomics of human-system interaction - Part 210: Human-centred design for interactive systems. ISO 9241-210.

Lieberman, H., Paternò, F., Klann, M., & Wulf, V. (2006). *End-user development*. Dordrecht: Springer.

Maeda, J. (2006) *The laws of simplicity*. MIT press.

MacLean, A., Carter, K., Lövstrand, L., & Moran, T. (1990). User-tailorable systems: Pressing the issues with buttons. *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 175-182). ACM.

Marshall, A., Stanton, N., Young, M., Salmon, P., Harris, D., Demagalski, J., Dekker, S. (2003). *Development of the Human Error template: A new methodology for assessing design induced errors on aircraft flight decks. Final Report of the ERRORPRED Project.* London.

Marvin, R. (2018). The best low-code development platforms for 2019. Retrieved April 30, 2019, from https://www.pcmag.com/roundup/353252/the-best-low-code-development-platforms

Mayer, R. E. (1981). The psychology of how novices learn computer programming. *ACM Computing Surveys (CSUR)*, *13*(1), 121-141.

Moran, T. P. (1983). Getting into a system: External-Internal task mapping analysis. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, (December), 45–49.

Myers, B. A., Pane, J. F., & Ko, A. (2004). Natural programming languages and environments. *Communications of the ACM*, *47*(9), 47.

Nielsen, J. (1994). *Usability engineering*. Elsevier.

Oppermann, R. (1994). *Adaptive user support: Ergonomic design of manually and automatically adaptable software* (pp. 1-13). CRC Press.

Paternò, F., & Wulf, V. (2017). *New Perspectives in End-User Development*. Springer.

Powers, D. (2011). Evaluation: from Precision, Recall and F-measure to ROC, Informedness, Markedness and Correlation. *Journal of Machine Learning Technologies*, 2(1), 37-63.

Repenning, A., & Ioannidou, A. (2006). What makes end-user development tick? 13 Design guidelines. *End user development* (pp. 51–85). Dordrecht: Springer.

Sajaniemi, J. (2008). Guest Editor's Introduction: Psychology of Programming: Looking into programmers heads. *Human Technology: An Interdisciplinary Journal on Humans in ICT Environments 4*(1), 4–8.

Silva, C. C. (2013). Audiovisual perception in a virtual world: An application of human-computer interaction evaluation to the development of immersive environments. *Proceedings of the 5th ACM SIGCHI symposium on Engineering Interactive Computing Systems (EICS 2013)* (pp. 175-178).

Stanton, N. A., Hedge, A., Brookhuis, K., Salas, E. & Hendrick, H.W. (2004). *Handbook of human factors and ergonomics methods*. Florida: CRC Press.

Tharwat, A. (2018). Classification assessment methods. *Applied Computing and Informatics*.

Young, R. M. (1981). The machine inside the machine: Users' models of pocket calculators. *International Journal of Man-Machine Studies*, *15*(1), 51–85.

Carlos César Loureiro Silva is the Research and Development Coordinator of the Perception, Interaction and Usability group at CCG - *Centro de Computação Gráfica*. He holds an MSc in Experimental Psychology from the University of Minho (Portugal, 2011) and a PhD degree in Informatics from the University of Minho (Portugal, 2019).

Joana Catarina Fernandes Vieira is a Usability Analyst and Researcher at CCG - *Centro de Computação Gráfica*. She holds an MSc in Experimental Psychology from the University of Minho (Portugal, 2008) and is concluding a PhD in Ergonomics from the University of Lisbon.

José Francisco Creissac Freitas de Campos is an Auxiliary Professor at the Department of Informatics of the University of Minho and a Senior researcher at HASLab/INESC TEC. He holds a PhD. degree in Computer Science from the University of York (UK, 2001).

Rui Miguel Silva Couto is a Senior Researcher at the HASLab/INESC TEC & University of Minho. He holds a PhD. degree in Informatics from the University of Minho (Portugal, 2017).

António Manuel Nestor Ribeiro is an Auxiliary Professor at the Department of Informatics of the University of Minho and a Senior researcher at HASLab/INESC TEC. He holds a PhD. degree in Informatics from the University of Minho (Portugal, 2008).