

Optimizing the Usability of User Interfaces based on Multi-objective Evolutionary Algorithms

Marwa Hentati¹ ^a, Abdelwaheb Trabelsi² ^b and Adel Mahfoudhi³ ^c

¹National School of Engineering, Sfax University, Sfax, Tunisia

²College of Computing and Informatics, Electronic University, Dammam, Saudi Arabia

³College of Computers and Information Technology, Taif University, Taif, Saudi Arabia


Keywords: User Interface, Optimization, Usability, Evolutionary Algorithm, Model-Driven Engineering.


Abstract: Solving the software system problems using optimization algorithms stands for an intrinsic area of research whose aim is to find an optimal solution according to a set of conflicting objectives. One of the most prominent problems is optimizing the software quality such as usability of user interfaces following the model-driven engineering (MDE). One of the main challenges of MDE process is identifying the highly-usable model according to a set of desired usability aspects. Although models may be equivalent from the functional viewpoint, they may differ from the non-functional perspectives. Besides, they do not fulfil the same usability properties. In this context, we addressed this issue by combining the power of model engine and the optimization algorithms. In this study, we propose to integrate a multi-objective evolutionary algorithm at the conceptual level of the MDE process. It allows to find an optimal (or near-optimal) model from a large search space according to a set of usability aspects and taking into account the context of use.


1 INTRODUCTION

Usability of the software systems is no longer a luxury, but rather an intrinsic factor of the success or failure of software systems (Seffah et al., 2006). Seeking to promote a highly usable user interfaces has triggered the appearance of some research studies taking into account the usability aspects of software systems (Abrahão and Insfran, 2017). In fact, a large number of these research works consider usability aspects only after the full implementation of systems. In this case, all changes introduced to the system design are costly in terms of resources and time. The basic objective of other research works resides in enhancing usability in the intermediate artefacts. Since the traceability between these artefacts and the implemented system is not well defined, taking into account the usability aspects at the conceptual level may not ensure the usability of the final system (Abrahão and Insfran, 2017). Supporting the model-driven engineering (MDE) process may alleviate these problems owing to its intrinsic traceability mechanisms (da Silva,

2015). Among the most outstanding research works in this context, we mention the Cameleon Reference Framework (CRF)(Calvary et al., 2003). It organizes the user interface (UI) development process into four levels of abstraction: At the highest level, the task & concept model (T&C) gathers the logical activities (tasks). Then, the abstract user interface (AUI) model groups a set of presentation-units which are modality independent. After that, at the concrete user interface (CUI) model gathers a set of computing-platform dependent interactors. Finally, the final user interface (FUI) model expresses the UI in terms of implementation technology. From this perspective, we support the idea that improving usability at the conceptual level leads to improve the final user interface (Hentati et al., 2016a). Recently, several research works have centred around supporting some usability aspects from the early stages of MDE process (Iñiguez-Jarrín et al., 2020). Nevertheless, there are still certain deficiencies quoting in particular the lack of details about generating the target model with an optimal usability (Hentati et al., 2016b). Hence, the search-based techniques stand for an appropriate solution to settle the software engineering (SE) problems by exploring the search space. Multi-objective optimization algorithms, in particular the evolution-

^a  <https://orcid.org/0000-0003-3832-8602>

^b  <https://orcid.org/0000-0002-1416-6867>

^c  <https://orcid.org/0000-0001-9522-8099>

ary algorithms, are applied widely to the majority of the SE problems including requirements management (Pitangueira et al., 2015), testing (Afzal et al., 2009) and maintenance (Bavota et al., 2014) disregarding the usability aspects (Harman and Jones, 2001).

The remaining of this paper is organised as follows. Section 2 displays an overview of related works. Section 3 exhibits the proposed process to optimize the usability at the conceptual level. Section 4 introduces a case study illustrating the feasibility and the obtained results. Finally, Section 5 wraps up the conclusion and provides new perspectives for future research.

2 RELATED WORKS

The central focus of this section is to summarize the state of the art works and the efforts performed in the field of software engineering via MDE approach. We are basically interested in some proposals that are considered relevant owing to their weight in related works.

Authors in (Mkaouer et al., 2020) proposed an approach for solving the SE refactoring problems. This problem is formulated as a multi-objective problem using the NSGA-III algorithms. The basic target resides in finding the optimal refactoring solution referring to a set of quality attributes. In the same context, authors in (Ouni et al., 2013) suggested to reformulate this problem as a single-objective optimization using an evolutionary algorithm to search the optimal sequence of refactoring operations. The operations quality is improved by minimizing the number of code smells.

Recently, several studies have addressed the Search-Based Model Transformation (SBMT) subfield which combines the search techniques and the model transformation engines (Boussaïd et al., 2017). We state, as an example, the MOMoT approach (Marrying Optimization and model Transformations) (Fleck et al., 2017). The basic objective is to transform a component model from a class diagram using optimization techniques by means of a Multi-Objective Evolutionary Algorithm (MOEA). To optimize the quality of the obtained model, authors attempted to minimize coupling and to maximize cohesion as objective functions.

Since the small screens devices are unceasingly used, an approach is set forward in (Raneburger et al., 2015) aiming to consider the generation of the graphical user interface from the Discourse-based communication model as an optimization problem. This approach rests on the branch-and-bound algorithm for

transforming the communication model into the structural model by selecting the optimal rule transformation. To accomplish this goal, three objective functions are formulated corresponding to the maximization of the use of available space, minimization of the number of navigation steps and minimization of scrolling the user interface.

The analysis of the above mentioned studies indicates that there are still such drawbacks to be overcome as considering the functional properties. Most of the proposed approaches search the optimal solution disregarding the usability optimization problem. In order to respond to this need, we opted for the optimization of the usability of user interfaces at the conceptual level following the MDE process by means of a multi-objective evolutionary algorithm.

3 USABILITY OPTIMIZATION PROCESS USING AN EVOLUTIONARY ALGORITHM

The basic goal of this research is to generate a concrete user interface (CUI) model with an optimal usability from an AUI model. We addressed the transformation of the AUI to the CUI as a usability optimization problem. As the usability displays a contradictory effect, we opted for multi-objective evolutionary algorithm. In prior research studies (Hentati et al., 2018) (Hentati et al., 2019), we tackled the generation of CUIs considering all alternatives related to abstract components which determine the search space.

Even through evolutionary algorithms (EAs) are generally based on the same generic framework, several requirements need to be taken into account to settle the usability optimization problem. The elaborated algorithm begins with the generation of an initial CUI that is added to the current population (Algorithm 1, line 10). For each iteration, all CUIs in the current population are assessed depending on the proposed objective functions (Algorithm 1, line 14).

One of the basic mechanisms of the evolutionary algorithm is to generate new solutions by means of the reproduction operators: the mutation operator (Algorithm 3) and the crossover operator (Algorithm 4). After i iterations, the search ends and returns a set of optimal solutions which stand for the approximate pareto front (PF_{approx}).

In order to select the optimal solution from PF_{approx} , we propose to use the *Knee point strategy* (Bechikh et al., 2011) depicted in Algorithm 2. The Knee point solution is a vector composed of the best objective values among the solutions produced

Algorithm 1: The evolutionary algorithm for the usability optimization problem.

```

1 Input
2 Population size :  $\alpha$ 
3 Mutation rate:  $\beta$ 
4 Crossover rate:  $\sigma$ 
5 Elitism rate:  $\gamma$ 
6 Iterations number:  $\delta$ 
7 Output  $PF_{Approx}$  Begin Generate the initial CUI ;
8 POP  $\leftarrow$  initial CUI;
9 Iteration number  $i \leftarrow 1$  ;
10 while  $i < \delta$  do
11     /* Objective functions*/ Evaluate
    the usability of each CUI in POP (using
    objective functions 1, 3 and 6);
12     /* Select best solutions*/
    Select  $(\alpha * \gamma)$  best CUI in POP ;
13     Add selected CUI in population POP';
14     /* Eliminate unfeasible CUI */
    foreach CUI in POP' do
15         if constraints are not satisfied
        (equations 9 and 11) then
16             Delete CUI from POP';
17         end
18     end
19 end
20     /* Mutation */ foreach  $(\alpha * \beta)$  CUI
    do
21         Apply mutation operator (Algorithm
        3);
22         Add CUImut in POP';
23     end
24 end
25     /* Crossover */
26     Select randomly  $(\alpha * \sigma)$  CUI from POP'
    for crossover;
27     foreach  $(\alpha * \sigma)$  CUI do
28         Select two CUIs as parents;
29         Apply crossover operator (Algorithm
        4);
30         Add crossed CUIs in POP' ;
31     end
32 end
33     Increment the iteration number  $i \leftarrow i + 1$ 
34 end
    
```

in each iteration. To find the optimal trade-off, these solutions are assessed using the trade-off worthiness metric introduced by (Rachmawati and Srinivasan, 2009). The closest one to the obtained 'Knee point solution' is then chosen as the final optimal solution.

Algorithm 2: 'Knee point' strategy.

```

1 Input Aproximatif Pareto front  $PF_{Approx}$ 
    (Output Algorithme 1)
2 Output Optimal CUI
3 Begin
4 foreach Objective  $O_y$  do
5     Select best values  $V_i$  relative to  $O_y$  to
    determine the knee point solution
    CUIKP ;
6 end
7 foreach CUI $i$  in  $PF_{approx}$  do
8     Calculate the distance between CUI $i$  and
    CUIKP(equation 12) ;
9 end
10 Select the optimal CUI;
11 End
    
```

3.1 Objective Functions

The usability optimization problem includes searching for the most usable CUI among a set of candidates, which corresponds to a large search space. In this context, we propose to search the CUI with an optimal usability according to three objective functions. Each of them stands for a usability sub-characteristic which can be either maximized or minimized. These usability sub-characteristics are adopted from an agreed usability model proposed by (Ammar et al., 2016). Indeed, this model is extended from the ISI/IEC 25000 standard to evaluate the usability at the early stage of the development process (ISO, 2005). The suggested objective functions are portrayed below:

3.1.1 Maximizing the Learnability

Optimizing the usability necessitates maximizing the value of the learnability sub-characteristic as identified in equation 1.

$$f_1 = \text{Maximize}(PR) \quad (1)$$

In fact, the learnability is specified by the 'Prompting' attribute. Equation 2 depicts the PR metric associated with the prompting attribute.

$$PR = \frac{\sum_{i=1}^n \text{SuppInfo}(x_i)}{n} \quad (2)$$

where PR is the proportion of labels that exhibits supplementary information, x_i is a label in a UI, n is the number of text fields that needs a specific structure and $\text{SuppInfo} = 1$ if label presents supplementary information, 0 otherwise.

3.1.2 Minimizing the Understandability

As for the Understandability sub-characteristic, it is characterized by a set of attributes having a relationship with cognitive load ¹. Hence, minimizing the value of understandability yields the optimization of the CUIs usability. In fact, we suggest minimizing two attributes, namely 'Breavity' and 'Information density', equation 3.

$$f_2 = Minimize(BR, ID) \quad (3)$$

BR and ID metrics are determined respectively in equations 4 and 5.

$$BR = MinDistance(x, y) \quad (4)$$

where BR is the number of steps needed to move from x to y, $x, y \in (interface)$.

$$ID = \sum_{i=1}^n Element \quad (5)$$

where n denotes the total number of elements per UI.

3.1.3 Maximizing the Operability

The usability of concrete models is optimized through maximizing the Operability sub-characteristic grounded on two attributes, which are 'error prevention' and 'user control action', equation 6.

$$f_3 = Maximize(ERP, UCA) \quad (6)$$

EPR and UCA metrics are specified respectively in equations 7 and 8.

$$ERP = \frac{\sum_{i=1}^n list(x_i)}{n} \quad (7)$$

where ERP corresponds to proportion of primitive lists used for each input element with a set of enumerated values, x_i stands for an input element, n represents the set of input elements with enumerated values, $list = 1$ if the input element with enumerated values uses primitive list, $list = 0$ otherwise.

$$UCA = \frac{\sum_{i=1}^n x_i}{n} \quad (8)$$

where UCA expresses the proportion of user control actions including cancellability and undoability actions, x_i indicates a UI action element and n denotes the total number of elements per UI.

¹The cognitive load (or workload) is related to the human memory properties. (Chanquoy et al., 2007). According to (Bastien and Scapin, 1993), the higher the cognitive load is, the higher is the risk of errors, lateness, exhaustion, and dissatisfaction of the user during interaction with the system.

3.2 Optimization Constraints

The majority of optimization problems, restrictions are imposed referring to the problem specification. Therefore, the solution acceptability is guaranteed only after the satisfaction of the optimization constraints. Moreover, it is difficult to generate a user interface with optimal usability for all user's profiles and run on all platforms (Gajos et al., 2010).

In fact, taking into account the context of use is a basic purpose in the MDE process. We have taken into account the context of use defined by (Calvary et al., 2003) who introduced the context as the triplet <user, platform, environment>. Since the expressiveness of environment is confined to the conceptual level, we opted for the platform (considering the screen size) and the user (considering the user experience level). Feasible solutions are chosen for the next population only if they satisfy the proposed constraints:

$$\sum_i^n (\epsilon + a_i) \leq availableSpace \quad (9)$$

with:

$$availableSpace = X_{resolution} * Y_{resolution} \quad (10)$$

where a_i refers to the area of a component i of the CUI, ϵ indicates the minimum space that separates two components, and availableSpace represents the screen area of the device to be involved.

A user with limited experience (or novice user) needs to be well guided, informed and protected from errors during interaction with the user interface. In fact, usability attributes like prompting (PR), and error prevention (ERP) are related to the user, especially to his level of experience. As a matter of fact, we set forward an optimization constraint based upon the level of the user's experience. The target lies in filtering solutions where the average (AVG) values of PR and ERP are under a threshold (V_{UX}). The later is fixed by the 'conceptor' considering the experience level of the target population. The equation 11 displays the proposed optimization constraint relative to the user level experience:

$$AVG(PR, ERP) \geq V_{UX} \quad (11)$$

3.3 Reproduction Operators

Reproduction operators invade the research space through the progress of candidate solutions. The derivation of new CUIs is accomplished by the mutation and crossover operators. The production of descendants plays an crucial role in the convergence of evolutionary algorithms to a global optimum.

3.3.1 Mutation Operator

Instead of applying standard mutation operators that might not be thoroughly convenient for our problem, we foreground a personalized mutation operator depicted in Algorithm 3. This operator offers a kind of diversity within the population and permits the optimal solution to escape from local optimum.

We propose to mutate one or a set of elements of a current CUI by (1) *changing* a component to another alternative and/or (2) *adding* a component to the CUI. Changing components in the CUI is fulfilled by replacing them with alternative concrete elements. It is significant to point out that the number of elements to be mutated rests on the mutation probability P_m fixed before launching the mutation operator.

As for the addition of elements, we propose the addition of control elements, like "validate" or "cancel" in the CUI, after verifying that it is impossible to cancel or to validate information in it.

Algorithm 3: Mutation operator.

```

1 Input CUI for mutation
2 Mutation probability  $P_m$ 
3 Output mutated CUI
4 Begin
5 if no 'valid' element in CUI then
6 |   Add valid element in CUI;
7 end
8 if no 'cancel' element in CUI then
9 |   Add cancel element in CUI;
10 end
11  $S_{CUI} \leftarrow$  size of CUI;
12  $N_m$  is the number of elements to be mutated;
13  $N_m \leftarrow S_{CUI} * P_m$ ;
14 foreach ( $N_m$ ) element in CUI do
15 |   Randomly select a concrete component
16 |   Comp from CUI;
17 |   Mutate Comp with an alternative;
18 |   Generate a new CUI_mut ;
19 |   Add CUI_mut in the population;
20 end
21 End
    
```

3.3.2 Crossover Operator

The crossover operator is applied at the level of panels of two randomly chosen CUIs. It can guarantee that all elements, which are gathered in a panel, are migrated. Moreover, the "one-cut crossover" is chosen because the "multiple-cut crossover" can disturb optimal solutions and the algorithm can converge to sub-optimal solutions. The crossover operator is depicted in Algorithm 4.

Algorithm 4: Crossover operator.

```

1 Input Two CUIs as parents: CUI_P1 et
  CUI_P2
2 Output Two CUIs as childs: CUI_Ch1 et
  CUI_Ch2
3 Begin
4 Cut CUI_P1 in two parts CUI_P11 and
  CUI_P12 ;
5 Cut CUI_P2 in two parts CUI_P21 and
  CUI_P22 ;
6 Combine CUI_P11 with CUI_P22 as
  CUI_Ch1;
7 Combine CUI_P21 with CUI_P12 as
  CUI_Ch2;
8 Add CUI_Ch1 and CUI_Ch2 in the
  population;
9 End
    
```

4 AN ILLUSTRATIVE CASE STUDY

The main aim of this case study is to emphasize the feasibility of the proposed usability optimization process. This case stands for a BTTS (Buying Train Ticket System). This system runs on terminals of passengers (smart phone, tablet, laptop, etc.). The UI of this system needs to create a feeling of comfort and ease of use to upgrade the user satisfaction degree regardless of the computing device. Numerous tasks are involved in BTTS like How to travel, Buy tickets, Discount and benefits, Train times etc. We are basically concerned with the generation of the CUI of Buy tickets task.

4.1 Generation of the Initial CUI

To better clarify the workability of the proposed usability optimization process, the case study started with the transformation from an abstract model to a concrete model. This ordinary transformation did not consider the usability aspects in a specific context of use. This system is implemented in mobile device-platformes with small screens size, and interacted by novice users having a limited experience of using a buying systems in general and BTTS in particular. The ordinary transformation takes as input the AUI model and produces the initial CUI model. Yet, the generated CUI fulfilled their functional objectives disregarding the usability aspects.

Table 1: The obtained objective values.

Iter	f_1		f_2		f_3	
	min	max	min	max	min	max
1	0.375	0.375	0.452	0.452	0.476	0.476
10	0.389	0.672	0.452	0.452	0.277	0.476
20	0.389	0.618	0.447	0.452	0.334	0.561
30	0.477	0.584	0.423	0.561	0.399	0.561
40	0.277	0.573	0.387	0.611	0.408	0.533
50	0.290	0.583	0.415	0.611	0.412	0.601
60	0.402	0.573	0.415	0.611	0.427	0.624
70	0.453	0.588	0.384	0.629	0.508	0.682
80	0.521	0.592	0.384	0.638	0.513	0.747
90	0.577	0.603	0.362	0.638	0.513	0.853
100	0.601	0.622	0.374	0.638	0.513	0.871
110	0.601	0.725	0.362	0.647	0.622	0.851
120	0.639	0.752	0.362	0.647	0.648	0.851
126	0.647	0.753	0.362	0.647	0.652	0.851

4.2 Obtained Results of the Optimal Transformation

The findings of running the usability optimization process at AUI to CUI transformation are outlined in Table 1 which summarizes the objective functions values after each step of 10 iterations. We infer that the values of these objective functions are stable from the 126th iteration. This last iteration groups 8 CUIs which represent the approximate Pareto front (PF_{approx}). The values of each of these CUIs are illustrated in Table 2. The "min" and "max" values, highlighted in Table 1, denote respectively the lowest and highest values of the objective functions, for each CUI produced in a population. Departing from these values, the "Knee point" solution (CUI_{KP}) is identified resting on the best CUI for each objective function. Indeed, the "Knee point" solution is defined as: $\langle f_1=0.753; f_2=0.362; f_3=0.871 \rangle$.

In order to specify the final optimal CUI among the 8 CUIs of the PF_{approx} , we applied the Euclidean distance permitting to identify the closest CUIs to the "Knee point" solution as identified in equation 12.

$$d(CUI_i, CUI_{KP}) = \sqrt{\sum_{O_y=1}^n (CUI_i - CUI_{KP})^2} \quad (12)$$

After computing the distance of each CUI with the "Knee point" solution, we realized that CUI5 is the final optimal CUI, which has the following values: $\langle f_1=0.705; f_2=0.374; f_3=0.851 \rangle$.

Figure 1 exhibits the final optimal CUI generated by the proposed usability optimization process for mobile devices. In this context, small screens are not able to present all the information and the user needs to scroll in order to see the interface. In fact, the high information density has a negative effect on

Table 2: The CUIs values of PF_{approx} and their distances from "Knee point" solution.

PF_{approx}	f_1	f_2	f_3	Distance "Knee point"
IUC1	0.753	0.366	0.652	0.203
IUC2	0.753	0.647	0.804	0.292
IUC3	0.661	0.362	0.683	0.201
IUC4	0.753	0.381	0.655	0.254
IUC5	0.705	0.374	0.851	0.043
IUC6	0.659	0.362	0.804	0.128
IUC7	0.647	0.644	0.871	0.306
IUC8	0.661	0.647	0.804	0.303

the user's performance and eventually on the usability of the CUI. To settle this problem, the CUI elements are distributed into three windows (Figure 1 a, b and c). Consequently, the association of windows instead of panels may ensure the optimization of the information density attribute (ID).

Yet, the window association negatively influences the brevity attribute (BR) in view of the increasing number of steps required to accomplish the task. Since the CUI is distributed into three windows, two steps are required to carry out the task. Therefore, as the number of steps is less than 3 (an acceptable upper bound), we infer that the brevity attribute is optimized.

The PR attribute is materialized in the optimal CUI by adding supplementary information about the data entry format. Indeed, this information guides users as well as facilitates their interaction with the system, where a specific format is expected to be entered such as the format of the dates (DD/MM/YYYY).

At another level, users of the BTTS have to be protected against errors during data entry. In fact, to accomplish this target, input elements with a list of options are transformed into lists, such as the drop-down list (UIDropDownList), the radio button (UIRadioButton) the checkbox (UICheckBox) and the combo box (UIComboBox). The presence of these elements allows to have a CUI with an optimal error prevention (ERP).

Additionally, the optimization of the user control actions (UCA) attribute is materialized by adding control buttons such as the validation and cancellation button. Indeed, the "Valid" button enables the validation of data entered by the user. The "Cancel" control button enables the user to interrupt an action or a treatment in progress.

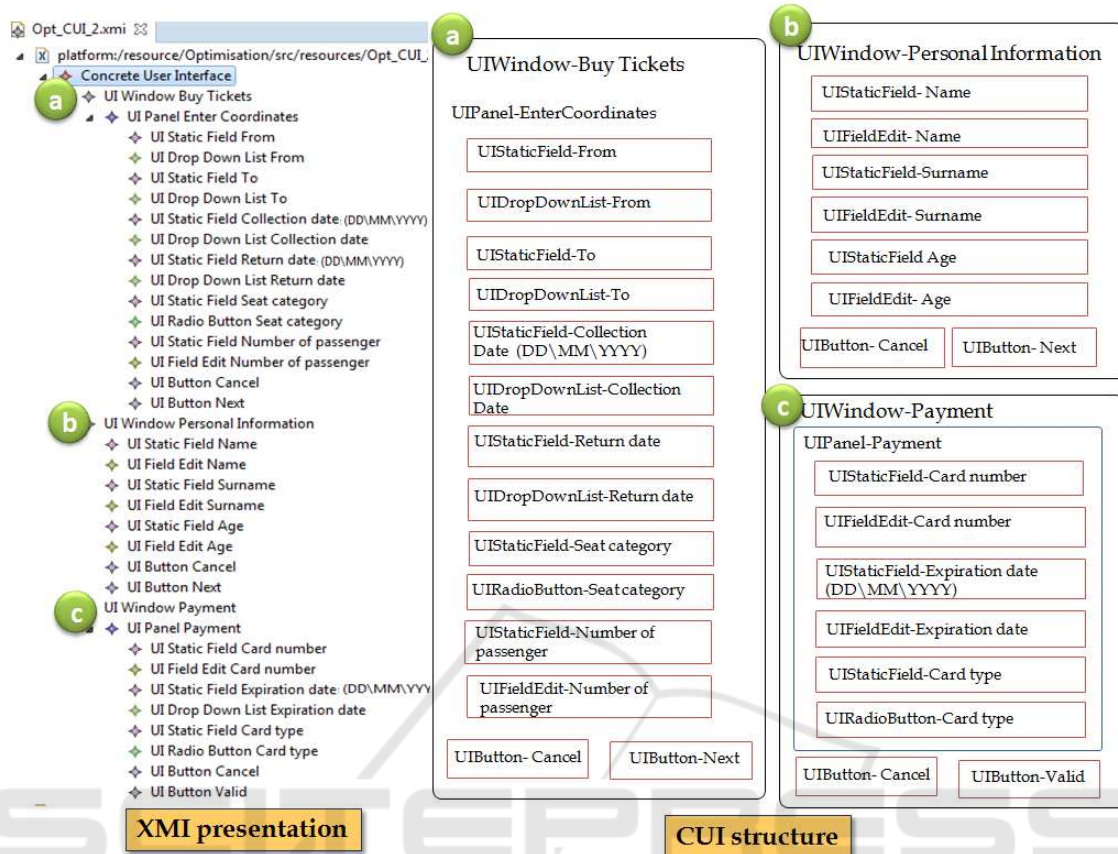


Figure 1: The optimal CUI in the mobile context.

4.3 Learned Lessons

The case study allows the additional assessment of the feasibility of optimizing the usability of user interface from conceptual models into MDE process. Furthermore, the suggested approach permits to take into account usability at an early stage of the development process. This may help avoid the usability problems once the system is operated. In addition, we inferred that the number of iterations as well as the performance of this process are affected not only by the optimization parameters but also by the quality of the initial CUI as well as the optimization constraints linked to the context of use. The found results can be regarded as promising. Indeed, some improvements can be introduced with respect to adding some usability attributes as well as validating them with an empirical study.

5 CONCLUSION

The paper elaborates a usability optimization process into MDE paradigm. This process yields three intrinsic steps: the first one concerns the generation of the initial CUI without taking into account the usability aspects. The second relates to the reproduction operator allowing to invade the search space and to derivate new CUIs. Only the best CUIs are chosen for the next population depending on the objective functions. We highlighted three objective functions based upon the usability sub-characteristics: learnability, undrestandability and operability.

This research may be extended, built upon and further elaborated as multiple studies may be undertaken in this area. This involves, in particular, adding more usability attributes to be optimized. Besides, to further enhance the performance of our optimization process, we propose to combine another optimization algorithm with the evolutionary algorithm. Indeed, this algorithm is not able to explore the whole search space by restricting a set of solutions. It would therefore be promising to combine the evolutionary algo-

rithm with a local search method.

REFERENCES

- Abrahão, S. and Insfran, E. (2017). Evaluating software architecture evaluation methods: An internal replication. In *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, pages 144–153. ACM.
- Afzal, W., Torkar, R., and Feldt, R. (2009). A systematic review of search-based testing for non-functional system properties. *Information and Software Technology*, 51(6):957–976.
- Ammar, L. B., Trabelsi, A., and Mahfoudhi, A. (2016). A model-driven approach for usability engineering of interactive systems. *Software Quality Journal*, 24(2):301–335.
- Bastien, J. C. and Scapin, D. L. (1993). *Ergonomic criteria for the evaluation of human-computer interfaces*. PhD thesis, Inria.
- Bavota, G., Di Penta, M., and Oliveto, R. (2014). Search based software maintenance: Methods and tools. In *Evolving software systems*, pages 103–137. Springer.
- Bechikh, S., Said, L. B., and Ghédira, K. (2011). Searching for knee regions of the pareto front using mobile reference points. *Soft Computing*, 15(9):1807–1823.
- Boussaid, I., Siarry, P., and Ahmed-Nacer, M. (2017). A survey on search-based model-driven engineering. *Automated Software Engineering*, 24(2):233–294.
- Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., and Vanderdonckt, J. (2003). A unifying reference framework for multi-target user interfaces. *Interacting with computers*, 15(3):299–308.
- Chanquoy, L., Tricot, A., and Sweller, J. (2007). *La charge cognitive: Théorie et applications*. Armand Colin.
- da Silva, A. R. (2015). Model-driven engineering: A survey supported by the unified conceptual model. *Computer Languages, Systems & Structures*, 43:139–155.
- Fleck, M., Troya, J., Kessentini, M., Wimmer, M., and Alkhazi, B. (2017). Model transformation modularization as a many-objective optimization problem. *IEEE Trans. Softw. Eng*, 43(11):1009–1032.
- Gajos, K. Z., Weld, D. S., and Wobbrock, J. O. (2010). Automatically generating personalized user interfaces with supple. *Artificial Intelligence*, 174(12):910–950.
- Harman, M. and Jones, B. F. (2001). Search-based software engineering. *Information and software Technology*, 43(14):833–839.
- Hentati, M., Ammar, L. B., Trabelsi, A., and Mahfoudhi, A. (2016a). An approach for incorporating the usability optimization process into the model transformation. In *International Conference on Intelligent Systems Design and Applications*, pages 879–888. Springer.
- Hentati, M., Ammar, L. B., Trabelsi, A., and Mahfoudhi, A. (2016b). Model-driven engineering for optimizing the usability of user interfaces. In *ICEIS 2016 - Proceedings of the 18th International Conference on Enterprise Information Systems, Volume 2, Rome, Italy, April 25-28, 2016*, pages 459–466.
- Hentati, M., Trabelsi, A., Ammar, L. B., and Mahfoudhi, A. (2018). Motuo: An approach for optimizing usability within model transformations. *Arabian Journal for Science and Engineering*, pages 1–17.
- Hentati, M., Trabelsi, A., Benammar, L., and Mahfoudhi, A. (2019). Search-based software engineering for optimising usability of user interfaces within model transformations. *IET Software*, 13(5):368–378.
- Íñiguez-Jarrín, C., Panach, J. I., and López, O. P. (2020). Improvement of usability in user interfaces for massive data analysis: an empirical study. *Multimedia Tools and Applications*, pages 1–32.
- ISO, I. (2005). Iec 25000 software and system engineering—software product quality requirements and evaluation (square)—guide to square. *International Organization for Standardization*, pages 3–20.
- Mkaouer, M. W., Kessentini, M., Shaout, A., Koligheu, P., Bechikh, S., Deb, K., and Ouni, A. (2020). Many-objective software modularization using nsga-iii. *arXiv preprint arXiv:2005.06510*.
- Ouni, A., Kessentini, M., Sahraoui, H., and Boukadoum, M. (2013). Maintainability defects detection and correction: a multi-objective approach. *Automated Software Engineering*, 20(1):47–79.
- Pitangueira, A. M., Maciel, R. S. P., and Barros, M. (2015). Software requirements selection and prioritization using sbse approaches: A systematic review and mapping of the literature. *Journal of Systems and Software*, 103:267–280.
- Rachmawati, L. and Srinivasan, D. (2009). Multiobjective evolutionary algorithm with controllable focus on the knees of the pareto front. *IEEE Transactions on Evolutionary Computation*, 13(4):810–824.
- Raneburger, D., Kaindl, H., and Popp, R. (2015). Strategies for automated gui tailoring for multiple devices. In *System Sciences (HICSS), 2015 48th Hawaii International Conference on*, pages 507–516. IEEE.
- Seffah, A., Donyace, M., Kline, R. B., and Padda, H. K. (2006). Usability measurement and metrics: A consolidated model. *Software Quality Journal*, 14(2):159–178.