**CHAPTER 1: INTRODUCTION**

In this chapter, we present the research context, problem statement and challenges, the research objectives, and thesis contributions that we have proposed to address these issues. We conclude with the structure of this report.

* 1. **Research context**

Adaptive user interface is a relatively new direction of research which has been increasing in the recent years (e.g. Gullà et al., 2011, Akiki et al., 2013, Macik et al., 2014). It considered as an important issue of research to improve human computer interaction.

Nowadays, human computer interactions become a complex task because software applications, devices, and user’s needs are diverse as well. Users were diverse by their skills, their fields of expertise, their experience and education level, their age, their tasks and goals, their motivation, and their intellectual and physical capabilities. When, user interface offered much functionality like menus, toolbars, dialogue boxes, etc. Also, devices were varying as well by the invention of computers, netbook, Smart phone, PDA, etc. So, the “one design fits all” approach becomes unable to support all these cases of variability in the context-of-use.

Therefore, AUIs have been promoted as a solution for context variability. It improves interaction between system and users and adapts itself to suit the context of use. The idea is to facilitate the user activity when using system. In fact, adaptation of user interfaces has become a necessity to facilitate the interaction between user and interface.

Adaptive user interface is defined as interactive interface that adapts their display according to the profile of user by monitoring human computer interaction (Rothrock et al., 2002). In other word, it changes its layout and rearranges the screen elements based on user needs. Adaptivity of interface is given by considering the context of use. Where, the context is modeling by three main dimensions: User (U), Platform (P), and Environment (E) (Bacha et al, 2011).

Many examples of AUIs were appeared in the literature during the recent years. Also, today there exist many established approach and frameworks for the design and implementation of AUIs. But, a few number of assessment approach was proposed to evaluate it. In fact, evaluation also considered as an important and challenging research issue in the area of AUIs.

* 1. **Problem statement**

In spite of the growing importance of adaptive user interface, they still exist critical issues in its development such as the lack of appropriate evaluation method. Evaluating AUIs need an appropriate method that assessing “dynamic” user interface. It aims to evaluate the adaptation of interface to their context of use. In fact, it is a complex task because it takes into account the variability of interface according to the current user, the platform, and the environment.

Proposed an automatic evaluation method to detect the problem of AUIs is a complex task and has many constraints and issues that should be considering:

* **Problem 1:** evaluating of AUIs need to take into account user characteristics. It is an important dimension that helps to verify the effectiveness of interface. But, users have a large number of characteristics such us age, motivation, education level, interest, gender, experience, etc.
* **Problem 2:** the variety of platform characteristics also should be considered in the evaluation method. For example each platform can be characterized by its screen size, operating system, screen type, memory, etc.
* **Problem 3:** also AUIs considering the variety of environment, it adapts their interface according to the current location, time, luminosity, etc.
* **Problem 4:** one other problem is that the adaptation of AUIs should be evaluated basing on some quality criterion. In literature there are many quality criteria as defined as the key of effectiveness of AUIs. For example, we quote these criteria: usability, guidance, compatibility, coherence, adaptability, learnability, etc. These criterions should used to evaluate AUIs and detect problem.
* **Problem 5:** another issue with AUIs research is that evaluation have focused on the benefit of adaptation, while problems have been ignored. And there is no consensus on the problem that caused negative impact on the effectiveness of AUIs.
* Difficulty to identify automatically the problem of adaptive user interface. the most of existing approach (Ref) have manually proposed evaluation rule basing on a set of evaluation metrics. But it is difficult to generalize this rules for all adaptive interface that are very different.
  1. **Research objectives and main contributions**

This master thesis seeks to evaluate adaptive interface and detect automatically its problem. The following sentence is the question of this research:

*How to evaluate adaptive user interface and detect its problem*

*basing on the context of use?*

While there are number of challenges to adaptive user interfaces development, we focus on those challenges that are pertaining to evaluate adaptive user interface and detect its problems.

* + 1. **Objectives**

Basing on, the later studies of evaluation techniques of user interface design for non-adaptive interface and the specification of AUIs that addresses to considering the context of uses, we aims to propose an automatic evaluation tool that allow to detect problems of adaptive interface, and to check if the adaptation decision making by the AUIs systems are meaningful and success to satisfy the special needs of users.

* + 1. **Contributions**

In order to accomplish the previous objectives, this master report has used to present an automatic method to detect the defect quality of adaptive interface that we propose the two following contributions:

1. To automate the detection of problem of AUIs we propose firstly a search-based approach using multi-objective evolutionary algorithms to automatically generate evaluation rules. These evaluation rules will used to detect problem. This first contribution aims to generate a set of evaluation rules based on combination of context, quality metrics, and problem.
2. The second contribution is considered on building an automatic tool to detect problem of AUIs. This tool is an Eclipse plug-in that takes as input the source code of adaptive system and generate as output the problems detected in its interface. This contribution divided into two parts. The first part is considered on adjustment process that update the threshold using in the evaluation rules according the measurement of quality metrics of current interfaces to be evaluated. The second part is considered on the detection of problem.
   1. **Report structure**

The rest of this report is divided into three chapters. In chapter 2, an overview of the state-of-art about adaptive user interface, and existing tools used to evaluate it is presented. Chapter 3 details the development of the optimization problem used to generate evaluation rules and the proposed tool that used to automate the problem detection process. Finally in chapter 4 we discuss the result of proposed solution and provide concluding remarks and perspectives.

1. **Related work**

In this master report an overview of the state-of-the-art on adaptation of user interface and evaluation of AUIs will be discussed. We will present a literature review about adaptation, and existing evaluation tools. In the survey of adaptation, we will start with introducing adaptation definitions. Second, we will present the two properties of adaptive user interface which are adaptivity and adaptability. Then we will present the different type of adaptation. Finally we will present the context of use and their importance for adaptation. In the review of evaluation tools, we will present the main definition of evaluation, and we will present existing tools used to evaluate the adaptive user interface.

## Adaptation of user interface

Researchers and specialists of Human-computer interaction give more importance to user interface layers from the beginning part of design process in application makes. They insist to design and modeling user, environment, and platform. Indeed, users interact with a computer system via a User Interface (UI) that should be easy to use, easy to understand, and support user’s needs.

* + 1. **Adaptation definition**

Adaptation of interface is characterized by its ability to make interface intelligent and personalized that allows it to detect the variability of the context of use. According to (Han et al., 1998) adaptation is defined as:

*“Adaptation is the process of selection, generation or modification of content (e.g., text, image, animation) to suit the user’s computing environment and context of use”*

The adaptation of user interface has been promoted to solve usability problems and to satisfy users needs and preferences, because it can be performed on the interface containers presentation such as layout, colors, sizes, and other design elements, and content like data, information, document (souii et al., 2015).

Adaptation has four different meaning, it depending on the level of adaptivity using by system (Paramythis., 2009):

* **Adaptation in general:** “*referring to the idea of having a system that can be adapted accordingly to the user’s need*”.
* **Self-adaptation:** “*adding the capability on the part of the system to perform the tailoring itself*”.
* **Adaptability:** “*incorporating the notion of the system’s being able to carry out by itself most of the steps required to decide upon and effect adaptation”*.
* **Adaptivity:** “*denoting, in addition to the above, that the system is capable of acquiring the user model, and performing non-trivial mapping between the contents of the said model and the range of possible forms of tailoring at runtime*”.

Moreover, user interface adaptation deals with systems adaptation that characterize by its sensibility of context of use. This kind of system subdivided into two types, *Adaptive* user interface, in which case the adaptation was determined in the design phase and the user cannot express any choice. And Adaptable user interface system, in which case the adaptation is only done if user has explicitly asked for it [Velsen et al., 2008]. An important distinction exists between an adaptive and an adaptable interface that we will detail in the following two subsections.

* + 1. **Adaptive User Interface**

Adaptive user interfaces (AUIs) are defined as “*systems that adapt their displays and available actions to the user’s current goals and abilities by monitoring user status, the system state and the current situation”* (Rothrock et al., 2002).

The adaptive user interface was implicitly adapted themselves. It allows system to adapt their structure, functionality or interface based on the user model generated from an implicit user input (beyond et al., 1987). According to (Gullà et al., 2011), adaptivity aims to facilitate the interaction between user and interface by using an adaptation strategy that can detect the current situation of user, environment, and platform and generate the appropriate interface at the run time.

Adaptive user interface seems to improve user interaction with systems by facilitating user task and performance, minimizing the need to request help, easing system use, helping users deal with complex systems, and avoiding cognitive overload problems. An implicit adaptation is acquired by using user’s profile modeling, advertisements, questionnaire, and survey. These techniques used in the design of user interface to make decision and adding an inference mechanism.

Users Data

User Model

System

Predictions or decisions about the user

User modeling or User model acquisition

Adaptation or user model application

Adaptation effect

Collect

Adaptation effect

**Figure 2.2**: Abstract system architecture for supporting adaptability

Today, many companies offer adaptivity on their web site such us Google which personalized advertisements, language, component adjusted to the user’s most recent search queries and according to their location (see Figure 2.3 for a screenshot of Google’s Swiss and Korean search engine site). Another example is Amazon Web site with its product recommendation mechanism. And Pandora application radio which personalize the choice of radio station. Windows also have some experience with adaptive user interface the most known example is Windows start menu that present a set of applications based on their relative frequencies of use to providing an easier access.



1. Google’s Swiss web site



1. Google’s Korean web site.

**Figure 2.3:** Google’s Web page in Switzerland and Korea

## Adaptability of user interface

Adaptable user interfaces are defined as “*Adaptable systems can alter aspects of their structure, functionality or interface on the basis of a user model generated from explicit user input, in order to accommodate the differing needs of individuals or groups of users and the changing needs of users over time”*( beyond et al., 1987).

Adaptable UI refer to the self-adaptation because it provides users the ability to modify and customize explicitly the interface according to their preferences. But, it is based also on knowledge that has been collected *dynamically* during past interaction process. Not all adaptation needs to be applied before interaction starts (at run time), adaptability provide users to select and change interface characteristics during all the interaction process. Also it often used to refer to customizability by the end user (Paramythis., 2009).

Adaptability gives users the ability to select the appropriate interaction way that they need from an alternative presentation and interaction characteristics through the selection of a specific user profile from a predefined list (Gullà et al., 2011).

User profile

And existing

Knowledge

Decision module

Adaptable

Interface

Knowledge

DB

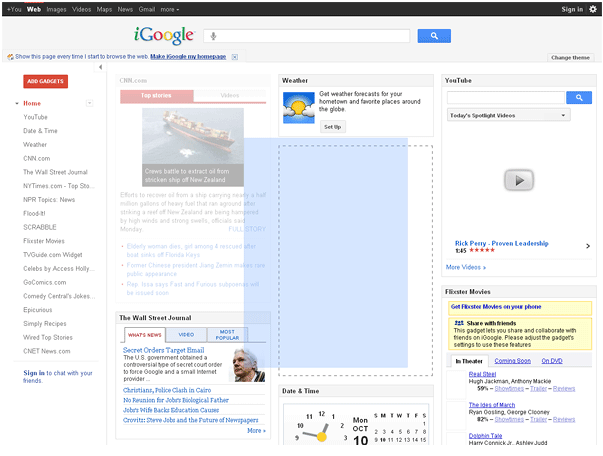
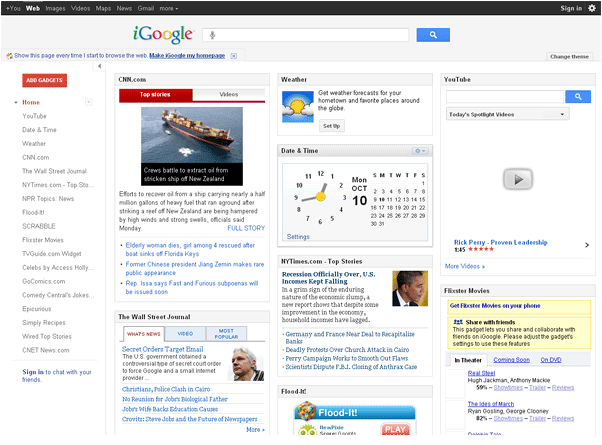
Adaptation effect

Monitoring

Component

**Figure 2.6:** Abstract system architecture for supporting adaptability

 iGoogleis an example of adaptable user interface, which give users the ability to modify the location and number of the components according to their needs. It gives users the ability to select their preferred content from list of gadget exist or add a new gadget, and to choose the location and size of components. See the screenshot below for a different components based on Google’s user research.

1. (b)

**Figure 2.7**: (a) and (b), is two screenshot for different Google’s research

* + 1. **Type of adaptation**

Adaptation can be classified in different type. For example, (Jameson, 2008) proposed a classification of adaption type. This classification is based on the function of adaptation. It divided into two supporting level:

* **Supporting system use**: which considering the routine tasks; adapting the interface; mediating interaction with the real world; giving advice about system use; controlling a dialogue (Paramythis., 2009).
* **Supporting information acquisition:** which helping users to find information; tailoring information presentation; recommending products; supporting collaboration; supporting learning (Paramythis., 2009).

In addition, (Knutov et al., 2009) present another type of adaptation. It has three types: content adaptation, adaptive presentation, and adaptive navigation. Each type of adaptation is characterized by using several techniques. Table 1.2 shows a summary of all type and their techniques:

* **Content adaptation:** it used to identify the content (data) most relevant to the users and the context of use. It filtering the content of interface to suit the user’s needs. It is applied to adapt the exchanged data with the user. This type of adaptation is achieved by fragment variant technique (Bunt et al., 2007). This technique allows selecting and combining an appropriate set of fragments, where each fragment typically corresponds to a self-contained information element (e.g. text, component, image, etc). Fragment variant proposed on many techniques: inserting fragment, removing fragment, altering fragment, dimming fragment, sorting fragment.
* **Adaptive presentation:** the task of adaptive presentation is to provide an effective interface that adapts its visualization. The effectiveness of interface is given by a successful presentation of layout, a suitable arrangement and size of component, etc (Pombinho et al., 2015). There many techniques used to provide an adaptive presentation such as stretch text, zoom, layout rearrangement, link ordering, link annotation, etc (Knutov et al., 2009).
* **Adaptive navigation:** this type of adaptation used to support user navigation in the interface. It aims to order user’s tasks and guide them. (Brusilovsky, 2007) proposed some techniques used to support personalized access to information of interface: direct guidance, adaptive link ordering, link hiding and removal, and adaptive link annotation:
  + **Direct guidance:** it suggest the “best next” task to do according to the user’s needs Brusilovsky, 2007).
  + **Link ordering:** it provides an adaptive ordering technology that prioritizes all the links of a particular page according to the user need and some user-valuable criteria. This technique helps to reduce navigation time (Brusilovsky, 2007).
  + **Link hiding:** it is used to restrict the navigation space by hiding, removing, or disabling links to irrelevant interface. Hiding protects users from the complexity of the whole user interface and reduces their cognitive overload (Brusilovsky, 2007).
  + **Adaptive link annotation:** This technology is used to augment the links with some form of annotation. It lets the user know more about the current state of the nodes behind the annotated links (Brusilovsky, 2007). There are many example of link annotation such as Manuel Excel introduced link annotation with different icons, ISIS-Tutor (Brusilovsky, 1994) changed the color and intensity of the anchors, and Hypadapter (Gauch et al 2007)explored altering anchor font sizes.

**Tableau 2.1**: Type of adaptation and their techniques

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Adaptation Type  Technique | | Content adaptation | Adaptive presentation | | Adaptive navigation |
| Inserting/ removing fragment |  | × |  |  |  |
| Altering fragment |  | × |  |  |  |
| Dimming fragment |  | × |  | × |  |
| Sorting fragment |  | × |  | × |  |
| Stretch text |  | × |  | × |  |
| Zoom/Scale | Conventional Scaling zooming | × |  | × | × |
| Fisheye view | × |  | × | × |
| Fragment summarization | × |  | × | × |
| Layout | Partitioning/ zooming |  |  | × | × |
| Rearrangement |  |  | × | × |
| Fitting in a template |  |  | × | × |
| Link sorting/ ordering |  |  |  | × | × |
| Link annotation |  |  |  | × | × |
| Combination techniques | Contextual links |  |  | × | × |
| Non-contextual links |  |  | × | × |
| Contents, Index links |  |  | × | × |
| Local and global Maps |  |  | × | × |
| Link Generation | Anchor adaptation |  |  |  | × |
| URL adaptation |  |  |  | × |
| Destination adaptation |  |  |  | × |
| Guidance | Direct Guidance |  |  |  | × |
| Global Guidance |  |  |  | × |
| Link hiding | Hiding |  |  |  | × |
| Disabling |  |  |  | × |
| Removal |  |  |  | × |

* + 1. **Contexte of Use**

Nowadays, software artifacts become more and more interactive, online information and devices are becoming large in size, and user’s number and needs are increasing as well. This diversity and increasing, provoke the importance of adaptability and flexibility of user interface, and the necessity to take into account the variety of platforms characteristics. So The main key of adaptation is its sensibility to the Context which is characterized by: User (U), Environment (E), and Platform (P), (Dey., 2001) (Dey & Abowd., 2000) define Context as:

*“Context is any information that can be used to characterize the situation of*  
*an entity. An entity is a person, place or object that is considered relevant to the*  
*interaction between a user and an application, including the user and the application*  
*themselves.”*

Recently, there are many studies that considered the context in the user interface design (e.g. Calvary et al., 2003; Vanderdonckt, 2005; Taconet and Aoul, 2008; Ayed et al., 2007; Hachani et al., 2009, Bacha et al., 2011). These studies based on the information collected about the three dimension of Context (User, Platform, Environment) to determine the way in which should present the information and customize the interface for different devices [Bacha et al., 2011].

Also, (Bacha et al., 2011) reported that it is crucial to consider the context modeling in the design of user interface, and they analyzed all the concepts and proprieties of the 18 proposals of context, and they classified them in categories of concepts according to their meaning and they organize categories around the three main context dimensions proposed for UI design: User Modeling, Platform modeling, and environment modeling.

The triplet (user, platform, and environment) forming the context-of-use can be  
considered as categories of aspects that could promote adaptive UI behavior (Akiki et al., 2015)

* + - 1. **User Modeling**

User modeling is defined as the representation of user characteristics which help to learn the profile of users by presenting them with a description of user’s attribute, age, gender, education level etc. (Bacha et al., 2011) divided user profile into five major categories that describe the user during its interaction with the platform: demographic information, contact information, user preferences, user state, and user abilities and proficiencies.

**Table 2.1**: The user profile categories [Bacha et al., 2011]

|  |  |
| --- | --- |
| Categories | Example |
| Demographic information | Name, gender, age, employment, etc |
| Preference | Culture, food, traveling, sport, etc |
| User state | Activities, physiological-state, etc |
| Ability | Skills, motivation, etc |
| Contact information | Telephone\_Number, e-mail, city, etc |

* + - 1. **Platform Modeling**

Platform modeling is generic way to characterize the platform and to describe the physical characteristics of the devices. According to [Bacha et al., 2011] platform divided into two categories:

* Hardware which composed on: memory, CPU, User interface characteristics, Network, (Demeure et al., 2009), (Aquino et al., 2010), (Dobre., 2011).
* And software that composed on: Operating system, application system, virtual machine, rendering engine (Dobre., 2011).
  + - 1. **Environment Modeling**

According to (Bacha et al., 2011) this model describes all information about the environment where the interaction takes place between the user and the platform. And most of the information related to this model are dynamic and can impact the content to be presented. Environment divided into three main classes: location, time, and environment condition (Pombinho et al., 2015).

**Table 2.3**:The environment categories

|  |  |
| --- | --- |
| Categories | Example |
| Location | City, country, GPS-coordinates, etc |
| Time | Extract time , symbolic time, event, etc |
| Environment condition | Weather, temperature, humidity, sound, light, noise, etc |

## Evaluation of adaptive user interface

Evaluation is widely considered as an important and challenging research area of adaptive user interface. However, evaluate the user interface by considering the context of uses and the human-computer interaction is a quite difficult task because the evaluation will require analysis of real users in a real context.

In this section we will present the definition and principle of evaluation, and the existing tools used to evaluate AUIs.

* + 1. **The evaluation: definitions and principles**

Researchers come from diverse scientific disciplines, which caused the diversity of adaptive user interface design model (e.g. user centered model, layered model, context awareness model). Each group of them design and evaluate user interface adaptation based on

their point of view. As a result, in the research field of adaptive user interface, there is no standard definition or evaluation method that has been shown to be applicable at a general level.

(Le Bodic., 2005) define evaluation as method which consists in the checking of adequacy between functional realization and the scenario of usage. Also, according to (Karat., 1997), evaluation is “ *the result of a process with a purpose in a context focused on an object*”.

So, evaluation of adaptive user interface has been used to compare user interface with their capacity to adapt to their real context of use. It allows determining the accuracy and adequacy between the interface and user’s preference, user’s need, platform used, and environment, in other word it allows to determine how accurately adaptation satisfies the needs of users. According to (Bellotti et al.2002) and (Arhippainen et al., 2003) the evaluation of adaptive system and their influence on users is quite difficult because the evaluation will require analysis of real users in a real context, and test users should have a fully operational, reliable and robust tool.

In literature there are few studies that present methods to evaluate adaptive user interface. These methods can be classified into three categories:

* Empirical evaluation: which characterized by using observation techniques in experiments, such as interviews, questionnaire, analyze of trace of use, etc. This type of evaluation used frequently to evaluate user interface because in this case we can used the final system or interface to observe and analyze the interaction between users and interface.
* Expert evaluation: “This type of evaluation based on the judgment of an expert in ergonomics or a specialist in human-machine interaction. It compares the performances and characteristics which are presented in the form of specifications, model or of prototype, to the standards or recommendations in order to detect a design errors” (Soui et al., 2012).
* Analytical evaluation: it is a theoretical approach that evaluates the design or code of interface. According to (Soui et al., 2012) we can classify analytical approaches into the following three categories: (1) the predictive models that are based on breaking down the tasks of potential users into users’ actions and users’ cognitive processes, (2) the automatic evaluation tools which analyze and verify the UI code and (3) the simulation methods (Feng et al. 2009).
  + 1. **Existing tools for adaptive user interface evaluation**

We present in this subsection, existing tools proposed to evaluate AUIs:

(Stephanidis et al., 1999) proposed an expert assessment approach to evaluate the adaptivity of *AVANTI Web browser* user interface. This approach provided two-fold assessment process. The first one, consist to assess the appropriateness of design interaction style based on user and usage characteristic. The second one, consist to evaluate the overall usability of the AVANTI information system. They evaluate: learnability, efficiency of use, memorability, errors, satisfaction, user attitude, and adaptability and adaptivity of the interface, by using qualitative and quantitative methods.

(Paramythis et al., 2001) proposed a new modular approach to evaluate AUIs. First it identifies the evaluation objects of AUIs called “modules”. These objects can be evaluating either separately or in combination. Then, they present the evaluation purpose. It considered on the decomposition of AUIs into modules and the subsequent assessment of these modules, based on specific criteria. Finally, they present the methods and techniques that can be employed for the evaluation of the different “modules”, in the different stages of the AUI development lifecycle (the evaluation process).

(McGrenere et al., 2002) proposed a comparison method to evaluate adaptive and adaptable interface. This method compare between: 1) a personalized interface which contain only features desired by users, 2) and the default interface which contain all the standard features. The study tested the effects of different interface designs on users’ satisfaction and their perceived ability to navigate, control, and learns the software. This method has shown that users prefer the personalized interface. But, it concluded that it is difficult to interpret this result, because the two user interface designs were present different functionality. And this result can be caused due to the difference in the complexity of the two interfaces, or to the fact that only one of them has an adaptive behavior.

In addition, in continuous research, (Findlater and McGrenere., 2004) shown study that compare also between static, customizable, and adaptive versions of split menus. From this study, they concluded that user-controlled customization is a feasible approach for personalizing UIs than system-controlled interfaces automatic adaptation. In a more recent study, (Findlater and Mcgrenere., 2008) have provided empirical evaluation that compare 36 different subjects of adaptive interfaces for small and desktop-sized screens, in other words they compare adequacy between interface and platform. They have shown that AUIs are more beneficial when screen real estate is constrained, and that the adaptive accuracy conditions were better in the small screen displays compared to the desktop sized displays.

(Tsandilas and Schraefel ., 2005) reported an empirical assessment to determine the effectiveness of user interface, they examined two adaptation techniques applied on list of textual selection. Also (Gajos et al. 2006) they reported two experiment in which they compare three adaptive users interface.

(Stober et al., 2010) proposed an automatic approach to evaluate adaptive user interface. They proposed to simulate different kind of structuring behaviors of users in order to evaluate the quality of the adaptation algorithm.

(Alemerien and Magel., 2014) they present GUIEvaluator which is a metric-based tool built to evaluate automatically the complexity of user interface based on five quantitative metric (Alignment, grouping, size, density, and balance), which are considered significant influences on interface usability.

Furthermore, Alemerien and Magel present another tool GUIExaminer (Alemerien and Magel., 2015) which support SLC metric (Screen Layout Cohesion). This metric used to predict the usability of user interface and it is considered as a hybrid metric because it is measured based on the structural, aesthetic, and semantic aspects of GUI layouts.

These last two tools represent an analytical evaluation that evaluate automatically “static” user interface, based on metrics calculation. In contrast, the evaluation of “dynamic” user interfaces lack of appropriate tools that evaluate automatically the adaptivity and

adaptability of interface. Another issue is considered in the studies that present a comparison methods between adaptive and non-adaptive user interface, the problem is obvious: the non-adaptive system may not have been designed ’optimally’ for the task. Today there are many established approaches and frameworks for the design and implementation of AUIs, but their evaluation is yet to be addressed in a comprehensive way.

* 1. **Conclusion**

In this chapter we describe a literature review about adaptive and adaptable user interface and its evaluation.

First we have reviewed the adaptation of user interface. We have started by presenting the definition of adaptation and their principle. Then we have presented their two main properties. Adaptivity which simplify and speed up the activity of user by an implicit adaptation. And adaptability which delegates the management of some interaction activities to users, and system plays a limited role of support by providing an explicit adaptation. And we have reviewed the appearance of context of use in the design of AUIs.

Then we have presented the main definition of evaluation and their principle. And we have presented some studies which reported evaluation methods for adaptive and adaptable user interface. Evaluating adaptive user interface means evaluate the ability of interface to adapt for diverse number of context of use, and implies making alternative evaluation decision, at various kind of the interaction.

In the next chapter we will introduce our contribution and method proposed to evaluate automatically the adaptive user interface and take into account the context of use in the assessment of interface.

1. **Adaptive User Interface evaluation By Objective**

This chapter presents our approach to evaluate AUIs. Our approach include into two contributions: a) generate evaluation rules, b) propose an automatic tool that detect problem of AUIs. In the first section, we introduce our approach overview. Second, we describe how we consider the evaluation of AUIs task as a multi-objective optimization problem. Then, we describe our proposed tool. First, we apply an adjustment mechanism to adjust the evaluation rules. Then, we describe *AUI\_Evaluator* plug-in Eclipse proposed to automate and generic the detection of problem.

* 1. **Approach overview**

We propose an approach that evaluates automatically AUIs. This method include of two steps. First, we started by generating evaluation rules. In fact, we consider this task as a multi-objective optimization problem that using combination of context, quality metrics, and problem to generate the evaluation rules. This approach takes as input a set of context (e.g., age, education level, interest, motivation, etc) and set of quality metrics (e.g., density, grouping, sequence, etc) and set of proposed problem that can be detected (e.g., loaded interface, disorder interface, low guidance, etc) , and it generate as output rule composed of two parts. The first one is the premise that includes an intersection between context and quality metric. While, the second one is the conclusion includes the problem. The approach uses base of example to evaluate the possible combination of rules.

Second, we proposed tool which can be used to detect automatically problems of an adaptive user interface. This tool use rules previously generated to detect the existing problem on the interface. This approach takes as input the interfaces to be evaluated and evaluation rules, and it generate as output detected problems. The tool proposed is an eclipse plug-in that automates a technique for the assessment of AUIs.

Our approach is based on multi-objective optimization problem which allows exploring a large search space and increasing the probability to generate an optimal solution that contains evaluation rules. Figure 3.1 illustrate our contribution.

Contexte

Metrics

Problem

Base of

example

**Metric Adjustment**

**Evaluation rules generation**

**1**

**Automatic detection of problem**

**AUI\_Evaluator**

Source code

MC900331015.WMFMC900331015.WMF

**Multi-objective**

**Search-based**

**Evaluation**

Problem

detected

Measure of quality metrics

**2**

**Figure 3.1:** Proposed approach.

* 1. **Multi-Objective Formulation**
     + 1. **NSGA-II principle**

During the last years, many Evolutionary Algorithms (EAs) were suggested to solve multi-objective optimization problems. Therefore, multi-objective optimization algorithm searching to generate a set of optimal solution or near-optimal solution in contrasts of mono-objective algorithm which looking for a single optimal solution. These set of near-optimal solution called also non-dominate solution and presented in the objective space with a curve called Pareto front.

Multi-objective optimization problem (MOP) is constrained by more than one objective which are typically a conflecting objective (Deb et al., 2002). These objective functions can be maximized or minimized.

In addition, MOP creates the condidate solution based on non-domination principale. When, the Pareto front is composed of the optimal solutions that doesn’t dominated by any others. So, the main keys of the multi-objective optimization problem are the dominance principle and the optimal Pareto front.

**Definition 1: dominance** is calculated by comparing two solutions. According to (Deb., 2011) solution *X* is said to dominate another solution if the both following condition are true:

1. The solution *X* is no worse than *Y* in all objective. The solutions are compared based on their objective function value.
2. The solution *X* is strictly better than *Y* in at least one objective

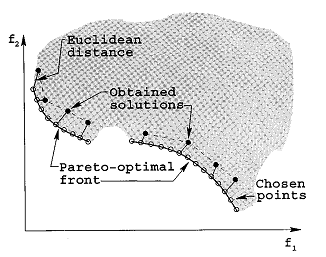
**Definition 2: optimal Pareto front** *“The Pareto-optimal front corresponds to global optimal  
solutions of several scalarized objectives”* (Deb., 2011).

Where:

PF\*: is a Pareto optima front.

F(x): is a given multi-objective problem.

P\*: is a set Pareto optimal generated by the MOP.



**Figure 3.2** multi-objective space representations

Several Multi-objective optimization problems were proposed in the literature, they aim to provide a well-convergence and well-diversity by searching many Pareto-optimal solutions (Kessentini et al., 2015). Many Multi-objective EAs has been suggested such us NSGA II. It is one of the most known algorithms that have been suggested to solve multi-objective problem. It is the second version of the “Non-dominated Sorting Genetic Algorithm” (Deb et al., 2002). NSGA II is characterized with:

* **The sorting non-dominated procedure:** in which the solution is sorted based on the level of non-domination into each front.
* **Implementing elitism:** it used to enhance the convergence properties of a multi-objective evolutionary algorithm because it stores all non-dominated solution. Elitism can speed up the performance of the GA and can help to keep the good solutions once they are found.
* **The crowding distance parameter:** it used to measure distance between each individual and their neighbors. This parameter enhances diversity and spread of solutions.
* **The constraints notion:** allows modifying the definition of dominance without the use of fitness functions.

NSGA-II uses an evolutionary process to solve problem that constrained by more than one objective which are typically conflecting objective. It provides to coverge toward a set of near-optimal solutions. The figure X show the basic principle of NSAG-II.

Non-dominated

Sorting

Crowding distance sorting

F4

F3

F2

F1

Offspring

population

Parent

population

Population

In next

generation

**Figure 3.3** NSGA-II procudure

* + 1. **NSGA-II adaptation**

We propose an approach that generates rules to evaluate AUIs. The evaluation of adaptive user interface by considering the context of use is a complex task, and has different constraints. For this reason we proposed to solve it as a multi-object optimization problem using NSGA-II. In this section, we are going to explain how we adopted NSGA-II to solve our problem.

As figure 4 shows, the algorithm takes as input the set of quality metrics, set of context criteria, and its equivalent of problems ones. From line 1 to line 5, the algorithm starts by creating the first parent population and the offspring population. Firstly, the population P0 is initialized randomly with a set of individual, the size of populations is fixed with Populationsize. Second, the first population is evaluated. Then, P0 is sorted with Fast-non-dominated-sort and the best parents will be selected. Then, the offspring population is generated.

NSGA-II uses Fast-non-dominated-sort algorithm to sort population based on non-domination principle. This algorithm compares each individual “*x*” with every other individual in the population until it is dominated by one of them. If no individual dominates it, the individual *x* will be considered non-dominated and will be selected by the NSGA-II to be a member of the Pareto front. All non-dominated individual constitute the optimal solutions of the problem (Pareto set).

Once the offspring population is created, all the solutions of this population would be evaluated using the two objectives function. Then, the offspring population is combined with the current parent population in new population called Union (line 8). Then, all the solution of this population will be sorted in order to return a list of non-dominated fronts F= {F1, F2, ...}, where F1 is the set of non-dominated solutions, F2 the set of solutions dominated only by solutions in F1, etc. The steps 12to 20 are performed repeatedly until all the solution will be sorted in their appropriate rank.

Once the non-dominated sort is completed the crowding distance calculation process is assigned, where NSGA-II uses a crowding-distance-assignment algorithm to calculate it. In fact, crowding distance of each solution is calculated by finding the Euclidian distance between each solution in the Pareto front Fi. Crowding distance parameters used to determine diversity, so in this step each solution assigned a diversity score.

Once the solutions are ranked based on non-domination and each one of them assigned a diversity score (crowding distance). Then, the solution that has the best rank would be emphasized. If a solution has the same rank, them which has a larger crowding distances would be emphasized. The steps 6 to 26 are performed repeatedly until some stopping criterion is fulfilled. Finally, the genetic operators (selection, mutation, crossover) are used to produce the set a new population. The output of the algorithm is the set of best solutions of evaluation rules.

**Input:** Set of metrics M.

**Input:** Set of context C.

**Input:** Set of problems P.  
**Output:** Set of evaluation rules.

1: Population  := InitializePopulation(PopulationSize , ProblemSize);  
2: Evaluate (Population);  
3: FastNondominatedSort(Population) ;  
4: Selected := BinaryTournament (Population, );  
5: Offspring :=  CrossoverAndMutation(Selected, Pcrossover, Pmuation );  
6: While (¬ StopCondition())

7: EvaluateAgainstObjectiveFunctions (Offspring);

8: Union:=  Merge(Population, Offspring) ;  
9:   Fronts := FastNondominatedSort(Union) ;  
10:  Parents := ø ;  
11: FrontL:= ø;  
12:   For (  Frontsi Front)  
13:       CrowdingDistanceAssignment(Frontsi)  
14:        If (Size(Parents)+Size(Frontsi) > PopulationSize )

15: FrontL := i;  
16:           Break()

17:        Else  
18:           Parents:=  Merge(Parents, Frontsi )  
19:       End   
20:    End For  
21:    If (Size(Parents)< PopulationSize)  
22:         FrontL := SortByRankAndDistance(FrontL);  
23:        For (P1 To PPopulationSize - SizeFrontL )  
24:            Parents := Pi  ;  
25:        End For  
26: End  
27:  Selected  BinaryTournament (Parents, PopulationSize  );  
28:  Population:=  Offspring;  
29:  Offspring  CrossoverAndMutation(Selected, Pcrossover, Pcrossover  );  
30: Return (Offspring);

Figure 3.2 High-level pseudo code for NSAG-II adaptation to our problem

* + 1. **Solution representation**

We aim to detect adaptation problem that can be appear in AUIs. To this end we propose to generate rules which would be used to detected the problems. So, our individuals should be composed with two parts. The first part (premise) represents an intersection of context and quality metrics which represent the condition. The second part, represents the detected problem. In our case an individual is a set of IF – THEN rules:

*IF (condition) THEN (conclusion: problem detected)*

Ri: IF **Ci = <H, L, M>**  AND  **Mi = <H, L, M>** THEN **Pi**

Ci: represent a characteristic of the context (user profile, platform, environment)

Mi: represent the quality metrics.

Pi: represent the problem

* + 1. **Creation of the initial population of solutions**

The initial population is created randomly with *N* population. The number of population is based on the problem range and our objective. Each population contains set of solution (individual) which has different size *S*. The size of solution determined randomly between Min-Solution-Size and Max-Solution-Size values.

* + 1. **Objective functions**

With NSGA II, after creating the solution we should evaluate it, to quantify its ability to solve the problem. Since we have two objectives to optimize, we are using the following two objective functions:

* ***Quality objective function (Q):*** used to evaluate the quality of solution. We considered the solution has high-quality if it detects the maximum of problem. So, this fitness function aims to maximizing the number of detected problem. In fact, maximizing the number of detecting problem corresponds to maximize the number of rule validating by the base of example. The function is normalized in the range [0, 1], and it calculated as follow:

n: size of solution.

To illustrate the fitness function, we consider base of example containing 10 interface evaluated manually by 250 users. Table 3.1 shows an example of the problem detected by solution X and the number of each problem in the trace.

**Table 3.1** solution example

|  |  |  |
| --- | --- | --- |
| **Problem** | **Solution X** | **Trace** |
| Workload AUI | 26 | 50 |
| Disorder AUI | 27 | 60 |
| Complex AUI | 49 | 60 |
| Irregular AUI | 0 | 40 |
| Low Guidance | 0 | 40 |

Thus, the solution X detects only 3 problems: workload AUI, disorder AUI, low Guidance, the fitness function of a solution X has:

* ***Size objective function(S)***: that evaluates the solution based on their size. A higher number of rules per solution do not necessarly mean that the result will be better. In other word, a small number of rules can be sufficient to provide good solution. So this function help to minimize the number of rules used to detect the problem. Minimizing the number of rule by solution correspond to minimize the size of individual (rule complexity) having also the optimal result in term of the first objective.

Therefore, these two objective functions used to generate solutions that has the minimum number of rules which able to detect the maximum number of problem.

* + 1. **Selection**

Once the solution is evaluated, the selection process of NSGA-II is carried out. This algorithm selected individuals by using binary tournament selection which is based on dominance and crowding distance. NSGA-II classifies solutions into different dominance levels. Then, it used comparison-operator to determine crowding distance to select potential individuals having the same dominance level.

* + 1. **Genetic operators**

To better explore the search space, we use this two genetique operator: crossover and mutation:

***Crossover :*** Crossover operator used to generate offspring. In our work we use single point crossover that selecting at random the crossover point and two parent solutions. First, crossover operator starts by selecting the two parents. Then it chooses the cut-point that used to split the parents into two parts. Finally, the two offspring was created, one them composed by the first part of the first parent and the second part of the second parent, the other one composed by the second one of the first parent and the first part of the socond parent. An example is shown in Figure 8

|  |
| --- |
| If (interest = H) and (Density >= 0.7)  then woklod AUI |
| If (education level = L) and (Regularity <= 0.2) then disorder AUI |
| If (Scren Size= L) and (Simplicity<=0.5)  then complex AUI |
| If (Use Experience = L) and(Density<=0.2)  then woklod AUI |

|  |
| --- |
| If (age = H) and (Density >= 0.7)  then woklod AUI |
| If (Interest = L) and (Simplicity <=0.2)  then disorder AUI |
| If (motivation = L) and (Grouping <=0.2)  then Low guidance |
| If (age = L) and (Regularity <=0.2)  then disorder AUI |

Parent 1 Parent 2

|  |
| --- |
| If (age = H) and (Density >= 0.7)  then woklod AUI |
| If (Interest = L) and (Simplicity <=0.2)  then disorder AUI |
| If (Scren Size= L) and (Simplicity <=0.5)  then complex AUI |
| If (Use Experience = L) and(Density<=0.2)  then woklod AUI |

|  |
| --- |
| If (age = H) and (Density >= 0.7)  then woklod AUI |
| If (Interest = L) and (Sequence <=0.2)  then disorder UI |
| If (motivation = L) and(Grouping<=0.2)  then Low guidance |
| If (age = L) and (Regularity <=0.2)  then disorder UI |

Child 1 Child 2

Figure 3.x Crossover operator

***Mutation:*** We use mutation operator to modify rule or to delete it from the set of rules. We modifing the rule by changing randomly the value of context, the value of quality metrics. An example is shown in Figure 8

|  |  |
| --- | --- |
| If (age = H) and (Density>= 0.7)  then woklod AUI | If (age = H) and (Density <= 0.4)  then woklod AUI |
| If (Interest = L) and (Simpicity <=0.2)  then disorder AUI | If (Interest = L) and (Sequence <=0.2)  then disorder AUI |
| If (Scren Size= L) and (Simplicity <=0.5)  then complex AUI | If (Scren Size= M) and ( Simplicit<=0.7)  then complex AUI |
| If (Use Experience = L) and (Density<=0.2)  then woklod AUI | If (Use Experience = L) and (Density<=0.2)  then woklod AUI |

Before Mutation After Mutation

Figure 3.x Mutation operator

* 1. **Problem detection**

In order to assess AUIs, we propose an automatic tool that detect quality problem. This section describes the problem detection approach. This approach is a generic mechanism for analyzing quality of interface by using evaluation rules. It propose to detect five different defect quality of AUI, namely complex UI, workload UI, disorder UI, low guidance, and irregular UI. The automation through the use of tools can help to reduce the error of quality metrics calculation and allow analyzing a large number of adaptive interfaces in a quick and repetitive manner. The detection problem process has two steps: 1) First, we need to adjust quality metrics of evaluation rules, 2) then, perform rules and detected the problems.

**Figure 3.3:** Problem detection approach

* + 1. **Adjustment of metric**

Quality problem detection has as input the pre-defined evaluation rules and the measurement of quality metrics of interfaces that should be evaluated. As highlighted in the previous section evaluation rules were composed on condition-part which is also composed on two parts context and quality metrics, and the conclusion-part which represent the problem.

The quality metrics are quantification mechanisms that support the examination of interface component characteristics. They are an important means used to achieve the identification of quality problems of AUIs. Thus, evaluation rules combining metrics with logical operators and thresholds:

IF (age = H) AND (**Density >= 0.8**) THEN Workload UI

IF (Screen Size= M) AND **(Simplicity <=0.3)** THEN Complex UI

In other words, a quality metric result can be interpreted as a certain symptom of one or more problems. In fact, in the detection process we need to compare the value of quality metrics of interface to be evaluated with an adequate threshold value. The threshold value is represented by evaluation rules.

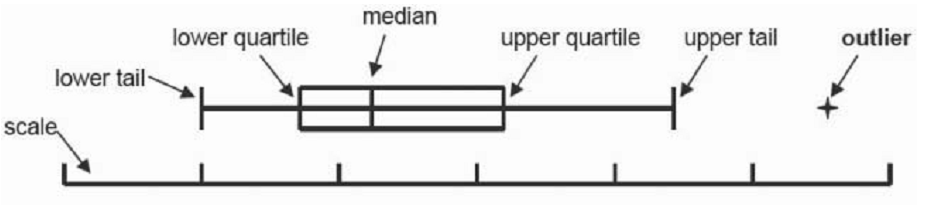
The pre-defined rules were generated automatically based on survey that describe the problem detected by users in three different applications. But to evaluate other interface we need to adjust the threshold of quality metric used in the evaluation rules. Each adaptive system has different number of interface and different number of component per interface.

Detection problem will vary depending on the selected value threshold. Increasing the value too much will cause more false negatives, while decreasing it in excess will cause more false positives. So, the use of quality metrics in the problem detection process needs an adjustment mechanism to improve accuracy of proposed tools.

In this context, we propose to filter the value of quality metric used in the evaluation rules. “*Filtering is a statistical means by which a subset of the measurement results is extracted based on the particular focus of the measurement, in the context of the detection strategy.*” (Marinescu et al., 2004).

In our work we used box-plot technique to adjust the threshold of quality metrics represented in evaluation rules. It is one significant example of a statistical means for detecting the abnormal values in a data set (Marinescu et Lanza., 2006).

This technique analyze threshold represented by evaluation rules. Box-plot take as input data set that analyze it to compute firstly median value. Then, use it to determine two pair of thresholds, which are the lower and upper quartile. These two thresholds will be used to update the measurement of quality metrics.



**Figure 3.3** Box-plot

* + 1. **Plugu-in: AUI\_Evaluator Tool**

In this subsection, we present an Eclipse plug-in *AUI\_Evaluator* that we proposed to detect problem of adaptive interface. The tool takes as input the adjusted evaluation rules and the measurement of quality metrics of different interface of system to be evaluated and generate as output list of detected problem of each AUI.

AUI\_Evaluator is tool detect the problem of AUIs based on the context of use and quality metrics. The metric quality used to measure the aesthetic aspects of GUIs (Alemerien, 2015). Figure X shows AUI\_Evaluator’s architecture which consist of the following component: UI data extractor, metrics adjustment, evaluation rule, metrics calculator

UI data

Source code

UI Data Extractor

Evaluation Rules

Metrics Calculator

Metrics adjustment

Evaluation

rules

Problem detected

1. **The UI data extractor** is the first task. It used to parse the UI developed in Java ans extract the structural and aesthetic attributes of each component on the interface.
2. **The metrics Calculator:** is used to calculate metrics measure. The metrics provided are:

* Density (D): used to determine the workload level of UI. High value of DM can caused workload UI problem.
* Regularity (R): used to calculate the regularity level of UI.
* Sequence (SQ): used to determine the sequence of component in the interface. Low value sequence can mean disorder UI.
* Grouping (G): used to determine the guidance level provide by the interface. Low value of GM can provides low guidance problem.
* Simplicity (SM): used to calculate the simplicity of interface. Low value of SMM can provoke complex UI problem.

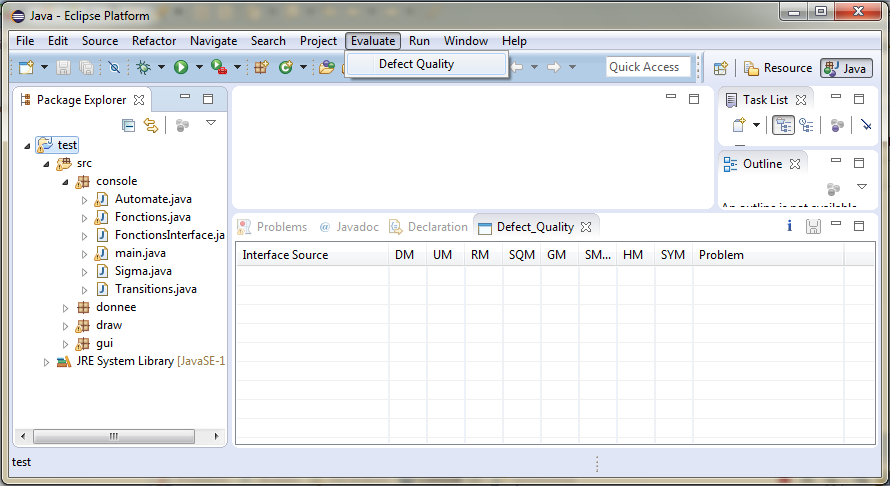
1. **Metrics adjustment:** is implemented to adjust the threshold of metrics basing on the set of metrics measurement generated by the metrics calculators.
2. **The evaluation rules:** are used to analyze the measurement of metrics with the current context in order to detect the problem of AUIs.

All the AUI\_Evaluator’s functionalities are presented thought an Eclipse plug-in. It can be utilized to evaluate interface and its problem under development or a part of running application.

AUI\_Evaluator extracts interfaces from the Java project. For each interface the corresponding context are read, and the corresponding metrics are calculated. Then, we performing the evaluation rules, there are two possible conditions:

* Evaluation rule use ≥ operator (greater than or equal to): In this case, if the measurement of quality metrics is above the defined threshold of evaluation rule then the problem is detected in this interface.
* Evaluation rule use ≤ operator (less than or equal to): In this case, if the measurement of quality metrics is below the defined threshold of evaluation rule then the problem is detected in this interface.

AUI\_Evaluator is a generic and automatic toll that can evaluate any adaptive interface. To use it, the user should imports the adaptive system under study as a Java Project and opens Navigator View in Java Perspective. Then, the user should select the “Evaluate” item in the menu bar and triggers the “Defect Quality” action, which in turn opens the corresponding view. After pressing the “Identify Problem” button the Defect\_Quality view lists the different interface with their quality metrics value and the set of detected problem, as shown in Figure 1.



Action

View

Identify problem

Adaptive System

Figure AUI\_Evaluator output showing identified problem

* 1. **Conclusion**

In this chapter, we present our two contributions of our research which are the proposed metrics of adaptive user interface evaluation and the meta-heuristic used to generate a set of evaluation rules that evaluate a several adaptive interfaces.

In the next chapter, we will discuss the experimentation of our evolutionary algorithm. Moreover, we will present a comparison study of our used multi-objective evolutionary algorithm with another mono-objective evolutionary algorithm.

**Chapter 4: Validation**

* 1. **Introduction**

Our study addresses three main research questions, which are defined below. We also explain how our experiments are designed to address them. The goal of the study is to evaluate the efficiency of our approach for generating correct detection rules while minimizing the rules-complexity and maximizing the number of detected interface defects.

* 1. **Research Questions**

We designed our experiment to answer the following four research questions:

**RQ1**: To what extent can the proposed approach enhance the graphical user interface quality by the detection of defects?

**RQ2**: How accurate is the proposed NSGA-II-based algorithm for defects detection?

**RQ3**: How does the proposed NSGA-II-based algorithm perform compared to a mono-objective approach?

To answer RQ1, we evaluate the performance of our algorithm adaptation on the detection of five different types of code-smells on several projects implemented by different developers coming from heterogeneous backgrounds (students, developers etc.). Four or the tested systems are senior design projects while the fifth one is an open source system.

To assess the accuracy of our approach, we answer RQ2 by computing two measures, precision and recall that are originally defined in the domain of information retrieval and widely used when comparing search algorithms results. When applying precision and recall in the context of our study, the precision denotes the fraction of correctly detected defects among the set of all detected code-smells:

The recall indicates the fraction of correctly detected code-smells among the set of all manually identified code-smells (that is, how many code-smells are undetected):

In general, the precision calculates the probability that a detected code-smell is correct, and the recall is the probability that an expected code-smell is detected. Thus, both values are in between 0 and 1, and the higher value the better it is. The expected rules were reported by the manual inspection of the interfaces.

To answer RQ3, we compare our algorithm to the mono-objective genetic algorithm and we report a comparison in terms of number of needed rules to detect the defects.

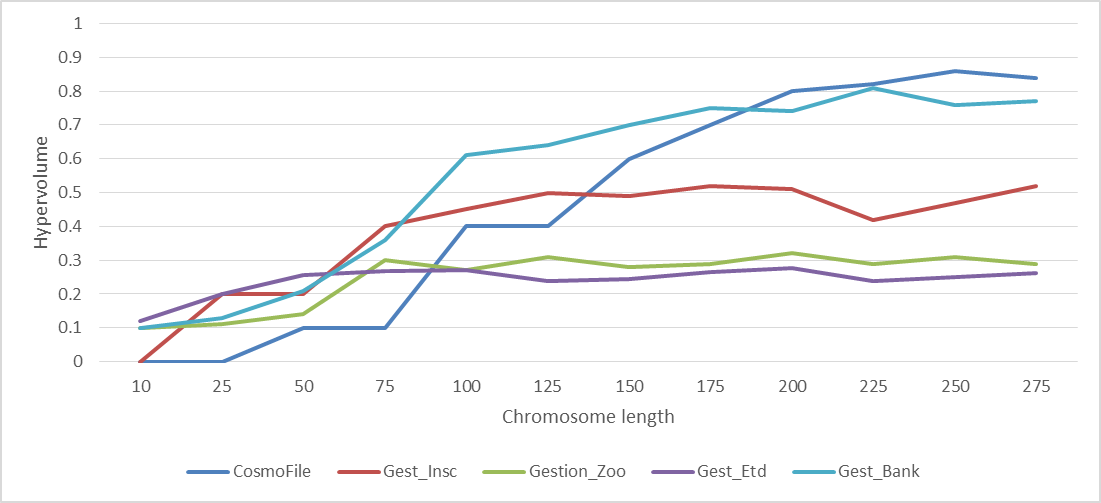
* 1. **Studied Project**

The validation is being conducted over the evolution of different open source Java GUI systems CosmoFile [1], Gest\_Insc [2], Gestion\_Zoo [3], Gest\_Etd [4], Gest\_Bank [5]. The corpus used includes releases of CosmoFile which is a great tool used to convert file to many different format. Gest-Insc is an application to manage the inscription of student. Zoo is application that serves to manage a graph which represents a world of animal. And, Gest-Etd is application that manages student information of National Institute of Applied Science and Technology. Finally, Gest-banck is application about banking management. Table X gives an overview about the studied systems. We have chosen these projects because of their small to medium size and because they contain multiple GUIs that can be used as input to our approach.

Table statistic of the studied system and solution length limit

|  |  |  |  |
| --- | --- | --- | --- |
| **System** | **Release** | **#Class** | **Solution length limit** |
| CosmoFile | v1.6 | 11 | [50, 200] |
| Gest\_Insc | v1.0 | 6 | [20, 100] |
| Gestion\_Zoo | v1.0 | 5 | [20, 80] |
| Gest\_Etd | v1.0 | 3 | [10, 30] |
| Gest\_Bank | v1.0 | 10 | [50, 200] |

As described in Table X, the upper and lower bounds on the solution length used in this work are set to dynamically and proportionally to the project size. Several search problems including defects detection are characterized by a varying chromosome length.



Figure

This issue is similar to the problem of bloat control in genetic programming where the goal is to identify the tree size limits. To solve this problem, we performed several trial and error experiments where we assess the average performance of NSGA-II using the HV (hypervolume) performance indicator while varying the size limits between 10 and 200 rules.

* 1. **Experimental Setting**

Since search algorithms are stochastic, they may generate various results for the same problem instance in multiple simulations. That’s why, our experimental study is performed throughout 31 independent runs for each problem instance and the obtained results are statistically analyzed by using the Wilcoxon rank sum test [reference] with a 99% confidence level (α = 1%). This guarantees that the obtained results of two algorithms used in this experiment are samples from continuous distributions with equal medians. For each input system, we compute the p-value obtained by comparing NSGA-II and mono-objective search results to determine whether the performance difference between them is statistically significant or just a random result. The following Table Y shows the tuning configuration for the two algorithms used in our experiments.

**Table 3.** The setting of common parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number of objective** | **Population size** | **Number of generation** | **Crossover rate** | **Mutation rate** |
| 2 | 100 | 5000 | 0.9 | 0.1 |

* 1. **Research Questions**
     1. **Results for Research Question 1**

In this section, we evaluate the performance of our NSGA-II adaptation on the detection of five different types of GUIs defects. Figure X summarizes our findings.

Figure 2…

The detection rules were able to identify various types of defects in the five input projects. This ability to identify different types of defects underlines a key strength to our approach. Most other existing tools and techniques rely on either detecting one kind of defect, and usually these techniques are manual, error prone and tedious. Other automated techniques rely heavily on the structural information of the GUIs. This is reasonable considering that some defects like the Complex UI are associated with a number of modules inside the interface. For defects like Irregular UI, however, the notion of number of modules and components is less important and this makes this type of anomaly hard to detect using only structural information.

Figure X reports the most frequent detected defect which are mainly Low Guidance and Workload UI, this gives an indication to developers to be more careful when designing GUIs as the preventive maintenance for the GUIs is cheaper in terms of budget and effort than the preemptive one. As the first one occurs on the design and implementation level while the latter one occurs after the software release to the client.

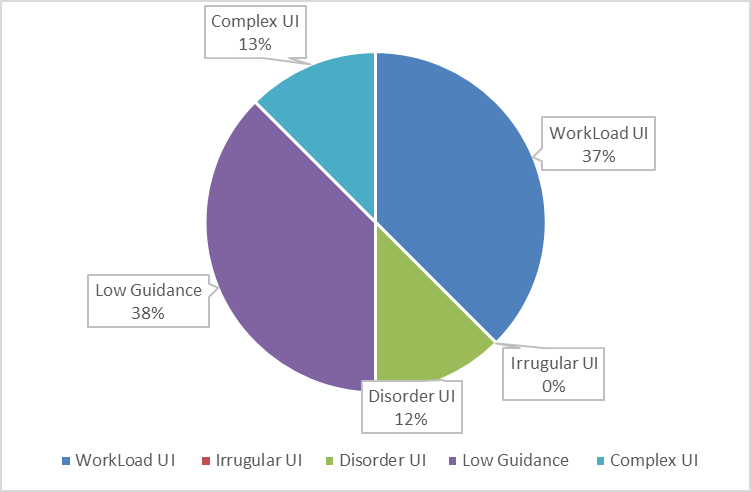
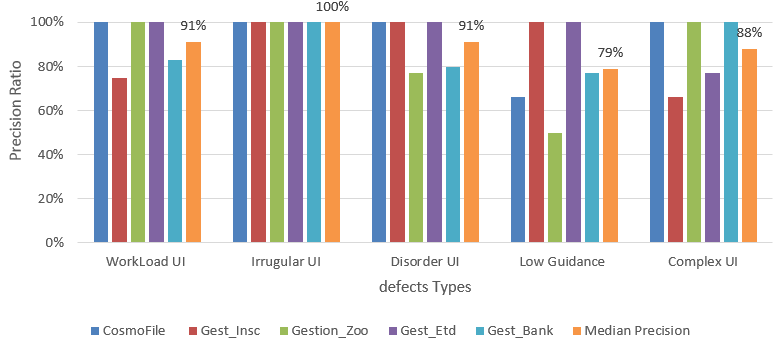


Figure 3

We noticed that our technique does not have a bias towards the detection of specific defect type. As described in Figure X, in all systems, we had relatively moderate distribution of each defects types. Having a relatively good distribution of defects is useful for a quality management as it allows developers to have more flexible scheduling when approaching their correction. Overall, all the five defects types are detected with good precision and recall scores in the different systems.

* + 1. **Results for Research Question 2**



Figure

The expected defects were detected with an average of more than 90% of precision and recall on the five tested systems. The highest precision for systems was found in Gestion\_Etd where 95% of defects were detected. The lowest precision was found in Gestion\_Zoo with a median of 78% of detected defects. This is can confirm that the list of returned defects did not contain high number of false positives and so our reported defects can be trusted by the developers and they can include it in their manual inspections process.

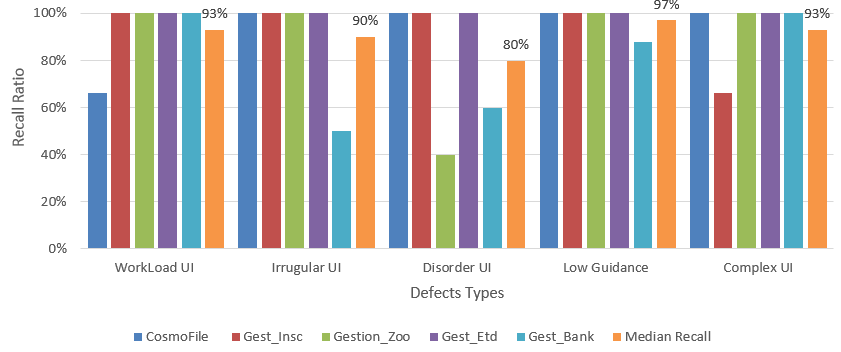


Figure 5

We found similar observation when analyzing the recall scores on the different interfaces where an average of more than 91% of expected defects were detected. The highest and lowest median recall ratios for the defects were respectively 97% for the low guidance and 80% for the Disorder UI. The PR and RE scores of Gest\_Bank (largest system) are among the highest ones with more than 90% which are better than the detection results of smaller system such as Gestion\_Zoo.

* + 1. **Results for Research Question 3**

To emphasize on the necessity of using a multi-objective technique, we empirically experiment the probability of defects detection when randomly generating a higher number of rules.

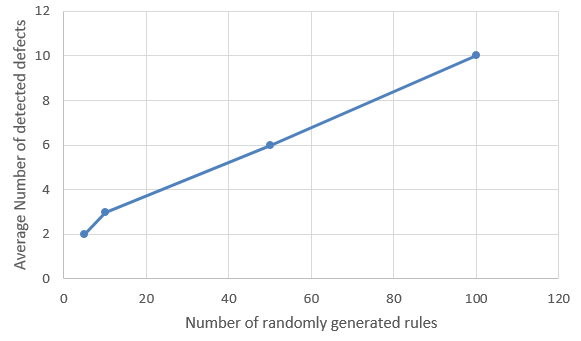


Figure X demonstrates that increasing the number of randomly generated rules increases the average number of detected defects over the 5 input projects. And so, reducing the number of rules, as part of reducing the solution complexity, reduces the chance of detecting defects. For that reason we consider the problem of defects detection while minimizing the number of rules is a multi-objective optimization problem.

