Generating Applications directly on the Mobile Device: an Empirical Evaluation

Andrea De Lucia, Rita Francese, Michele Risi, Genoveffa Tortora

University of Salerno, Via Ponte Don Melillo, Fisciano (SA)
{adelucia, francese, mrisi, tortora}@unisa.it

ABSTRACT
This paper presents an investigation, based on the combined use of two techniques: a questionnaire-based survey and an empirical analysis, to assess the effectiveness and efficacy of the MicroApp environment to support End-Users in the visual composition of their own applications directly on their mobile phone. The satisfaction of the End-Users has been investigated as well. The context of this study was constituted of students, administrative personnel and consultants of the University of Salerno. The survey shows a positive satisfaction degree of all the involved subjects, while the empirical analysis reveals that the use of the Micro App tool increases the efficiency and, in case of complex tasks, also the simplicity with respect to the use of a PC-based similar tool proposed by Google.

Categories and Subject Descriptors
H. [Information Systems]: Information Interfaces And Presentation - Evaluation/methodology, Graphical user interfaces (GUI)

General Terms
Experimentation, Human Factors, Languages.

Keywords
Empirical Study, End-User development; Mobile Applications; Visual Programming

1. INTRODUCTION
Today’s top-of-the-range mobile devices offer a variety of functionalities and the number of applications available for them grows day by day. Nevertheless, few of the available applications take into account that mobile users have different preferences and employ the applications in various situations. In many cases, it is impossible to foresee all the context of use and the actual requirements are often difficult to describe. One possible solution is to provide to the End-User the means to easily compose and customize mobile applications, starting from a set of available simpler features and services.

Some existing tools support the generation of mobile applications on the Personal Computer (PC) and successively download it on the mobile device, see [1][6] as an example. In some cases, they require the user to compose the application user interface [1]. The main idea of these tools is to avoid writing the source code. In particular, a mobile application requires the knowledge of different technologies, such as a particular programming language (i.e., Java or Objective-C), different software development kits related to different mobile devices (i.e., Android, iPhone and so on). This involves a high learning curve with a subsequent high effort for End-Users or novice programmers. In [4] we proposed MicroApp, an approach enabling an End-User to model a mobile application as a composition of pre-existing applications/services directly on the mobile device. In particular, the MicroApp environment supports the composition of an application by visually modeling it through graphical symbols, associated to a particular application behavior and to a specific user interface. By opportune connecting these graphical symbols, the user can describe complex behaviors. Mobile End-User development is at the beginning phase and presents new issues mainly due to the characteristics and the limitations of mobile device interfaces.

This paper presents a user study aiming at assessing the effectiveness and efficacy of the mobile MicroApp environment to support End-Users in the generation of mobile applications. The satisfaction of the End-User is also investigated. This study is based on the combined use of two techniques: a questionnaire-based survey and an empirical analysis. The empirical analysis has been conducted comparing the MicroApp environment with App Inventor [1], a visual tool proposed by Google for the generation of mobile applications. The context of this study was constituted of students, administrative personnel and consultants of the University of Salerno.

The remainder of the paper is organized as follows: Section 2 discusses related work concerning the usability of mobile applications, while Section 3 briefly summarizes the main characteristics of the evaluated tools, App Inventor and Micro App. Section 4 describes the design of the presented user study. Section 5 discusses the observed results and, finally, Section 6 concludes the paper.

2. RELATED WORK
Usability cannot be related to a precise aspect of a software system, but it strongly depends on the intended use of the system. According to the standard ISO 9241-11 [9], usability represents the degree of a software product to be used by specific users to
pursue specific goals within a specific usage context effectively, efficiently, and with satisfaction.

Usability methods have been developed for many years to evaluate the efficiency, interaction flexibility, interaction robustness, and quality of user interfaces. Several usability evaluation techniques and guidelines have been proposed in the literature aiming at planning and realizing usability studies. For example, Ricks and Arnoldy in [18] assert that all the usability studies follow the same basic steps although a wide variation of products can be analyzed (e.g., software applications, printers, web sites, etc.). Generally, usability produces several benefits ranging from the reduction of the training costs to the improvement of the user satisfaction. In particular, usability is often the major concern in the usage of mobile applications, due to small screens and lacking of a full-scale physical keyboard. A recent study shows that 57% of the users experienced the slow-to-load problem when using the mobile Web, and 48% of the users found mobile applications difficult to read and use.

Usability evaluation can be conducted following different methods that can be broadly divided in two groups: heuristic evaluations as rule-based evaluations performed by usability experts and user studies. Concerning the mobile application skills, a new category of users has been considered: the casual users, that is the users that do not use the device for working, and reveal high skills similar to the ones of the expert users. On the other hand, user studies are based on the direct involvement of users through observations, interviews, and questionnaires and include empirical methods and observational methods. These techniques are quite efficient for evaluating usability of user interfaces when concrete tasks are considered. In addition, these evaluations can be performed in laboratory settings as well as in field studies. In [2], Beck et al. proposed several usability testing techniques aiming at facilitate the systematic data collection in a controlled environment, considering, in particular, the identification of usability problems related to the mobile use. Their experience revealed that the subjects involved in techniques with much motion and navigation are more likely to miss a button on the interface. In many studies test subjects were required to walk while using the mobile system being evaluated. Paternò et al. in [17] proposed to exploit the user logs for recording interactions of the mobile user to be observed. In this way it is possible to perform remote usability studies of mobile devices. In our specific case, we adopted a traditional evaluation techniques in a laboratory setting, based on a standard questionnaire proposed in [10] by IBM, also valid in the context of App Inventor, the PC-based tool considered during the empirical evaluation as comparison term. Standardized satisfaction measurements offer the advantages of objectivity and quantification. Indeed, adopting them, practitioners can report detailed results more precisely than they could use only personal judgment. Because the usage of App Inventor requires the user to sit, we adopt the same condition for the evaluation of the MicroApp environment. Our intention is to offer the user the possibility of creating customized applications in whatever place, without the PC aid.

The interface of the MicroApp environment has been described in [5] together with further implementation details. A preliminary evaluation of the MicroApp environment has also been proposed in [5], where the data for a usability study have been gathered considering a group of ten volunteers. A single task concerning the creation of a MicroApp was proposed to the participants. Results were encouraging. In this paper we aim at understanding if the limitation of the mobile device, the development process and the editor features are still appealing when compared with the potentiality of a well known desktop application with the same aim.

3. THE MOBILE APPLICATION GENERATION TOOLS

In this section we briefly summarize the main characteristics of the tools examined during the evaluation. Further details on the two applications are available in [4] and [5], and at web page1.

3.1 The MicroApp environment

This tool aims at supporting an End-User in the creation of focused mobile applications, called MicroApps. An End-User generates and uses a MicroApp directly on the mobile phone. A MicroApp is designed by graphically composing the functionalities offered by the various phone applications, such as taking an image from the Camera object and saving it (i.e., the Camera.Take and Image.Save actions), retrieving the contacts list from the Contacts object (i.e., Contacts.List action), and sending an email using the Mail object (i.e., Mail.Send action), etc. Each action or service exposes a description of its user interface that enables to generate automatically the MicroApp user interface.

An ad-hoc developed mobile Visual Editor, designed considering the limited size of the device screen, supports the user in the modeling of the behavior of a MicroApp by composing its application logic. In particular, the approach adopted by the MicroApp editor allows the users to manipulate and connect the actions to build their mobile applications. Graphically, an application action available on the mobile device is represented by a rounded square containing the application icon, such as Camera and Mail, and the name of the action, such as Preview and Send, respectively. The input/output parameters are represented by colored bullets. In particular, as shown in Figure 1, the input parameters are depicted in the higher part of the square, whereas the output parameters are shown on the bottom.

Figure 1. Example of application actions

The parameters are differently colored, depending on the type of the corresponding object. As an example, the pink colored parameter represents an Image object, while the cyan colored parameter represents a Contact object, containing the contact data (e.g., name, surname, address, email, cellular phone and so on). Similarly, the red bullet represents a text string and, finally, the yellow bullet represents an email object.

1 http://appinventoredu.mit.edu/
A MicroApp is composed by dragging and dropping the action icons in a Composition Area, shown in Figure 2, where columns allow the user the sequential and parallel composition of actions. This editor supports only these types of composition rules, avoiding more complicated structures as cycles, because one of the usability objectives of the proposed approach is that the language has to be easy to use and to learn and target applications should be the result of a reduced number of compositions.

The user adds a new action by pressing the button represented by the circle icon in the middle lower part of the Composition Area. Then, the editor opens a new window showing a list of the available applications. The list of all available applications is provided by the MicroApp Repository. When the user selects an application (e.g., Camera, Contacts, Net and so on), the editor shows the list of the actions available for that application.

To compose this sentence, the user initially drags and drops the actions on a column (i.e., the first column). Starting from the leftmost higher part of the screen, the user first selects a static contact. This means that at design time the user has to choose a contact present in the contact list. If he does not choose the contact, it will be determined at run time. Then he adds the Contact.Preview action, showing on the device screen the contact image. In the second column, a Text.Static action enables the user to statically insert a text. When the call ends, this text is displayed on the user screen trough a dialog message. Let us note that the output parameter of the topmost action is compatible with input parameter of the lower action. Note also that if, as in the case of the second column, there is an empty space, the action Text.Static is automatically lengthened. Finally, a workflow engine enacts the composed process when the application is invoked. In particular, this engine, named MicroApp Engine, is a mobile application that manages and executes modeled MicroApp specifications.

3.2 App Inventor

App Inventor is an application proposed by Google allowing the End-User to create applications for the Android operating system [1]. It provides a graphical PC-based application that adopts a block editor to create simple programs to be deployed on the mobile device. In particular, an application is created using two tools:

- the App Inventor Designer tool, shown in Figure 3(a), where the components of the applications are selected; it runs in the web browser.
- the App Inventor Blocks Editor tool, where the behavior of each component is described by assembling the program

---

2 http://beta.appinventor.mit.edu/learn/tutorials/piccall/piccall.html

3 By the end of 2011 Google will also be making the complete App Inventor source code publicly available under an open source license, so that anyone can study the code and modify it as they desire. In order to ensure the future success of App Inventor, Google has funded the establishment of a Center for Mobile Learning at the MIT Media Lab.
blocks. Blocks are put together like pieces of a puzzle, as depicted in Figure 3(b) and (c).

Figure 3 shows the PicCall application visual composition (i.e., Task 1 in the experiment) created using App Inventor. First, by means of the Designer, the user puts a button (named TopButton) on the screen and defines its dimensions. Next, he/she adds a PhoneCall component (named TopCall) that is in charge to perform a phone call, and a button that enables the contact picker component (named TopPick) to choice a contact in the contacts list. Then, the user defines the behavior of the application using the Blocks Editor and selects the when TopButton.Click do block. In the do slot, the user places a call TopCall.MakePhoneCall block from the PhoneCall, that represents the procedure that is able to perform the phone call. The result is shown in Figure 3(b). Figure 3(c) depicts the description of the behavior of the application after pressing the "Pick a number" of the TopPick button. When it is pressed, it brings up the phone contacts list and lets the user pick someone. After pressed, the data associated to the contact, such as the phone number to be called and the contact picture are associated to the TopButton and to the TopCall components, respectively.

4. THE DESIGN OF THE USABILITY STUDY

In this section we present the design of the empirical study presented here. In particular, we describe the data set and then the techniques we have adopted to assess the efficacy, efficiency of the tool and the users’ satisfaction. The controlled experiment has been performed following the template suggested by Wohlin et al. in [21].

4.1 Experiment definition and context

The study was conducted in a research laboratory at the University of Salerno. Data for the study have been gathered considering a group of 24 volunteers. 10 of them were Bachelor students in Computer Science at the University of Salerno, while 4 of them belong to the administrative staff of the same university and 10 are employed in different organizations, i.e., consultants. The Bachelor students had programming experience and good knowledge of mobile phone devices. The BoxPlot in Figure 4 summarizes the subjects’ background concerning their skills in PROgramming (PRO), PC Knowledge (PCK), Android Mobile Device Usage (AMDU) e App Inventor Usage (AIU). Concerning the AMDU knowledge, we followed the directions proposed in [15] and examined, in particular, referred to the Android operating system, the participant knowledge concerning the capability of capturing and viewing audio/video, installing a game, browsing, using a search engine, using the Bluetooth file transfer and performing calendar entry tasks. We directly observed the participants while performing these tasks and classified the user experience level giving, for each accomplished task, a score according to the seven-point Likert scale [14]: from 1 (very high) to 7 (very low).

The data concerning PRO, PC and AIU Knowledge are collected during a pre-experiment questionnaire. The answers to the survey questionnaire have been evaluated on the seven-point Likert scale: from 1 (very high) to 7 (very low). As shown in Figure 4, half of subjects had programming experience, but most are very practical with the usage of Android mobile devices and have a good PC knowledge. Only one subject has a preliminary Knowledge of App Inventor.

![Figure 4. Subject background](image)

During the experiments, we assigned the following two tasks to each participant, selected from the tutorials proposed by App Inventor.

- **T1**: implementing and generating the mobile application PicCall. This application associates an image and a phone number to a button. These information are taken from the contacts list. When the user clicks on the image, the number is called.
- **T2**: implementing and generating the mobile application "No text while driving". This application allows the user to automatically send a message when the device receives a call. The response message is loaded and saved in a database.

Two different methods, namely MA and AI, have been considered to perform the selected generation tasks. Both methods aimed at generating the two applications T1 and T2. The difference is that MA is performed using the MicroApp generator, while AI is performed using App Inventor.

The tasks were expected to be accomplished within one hour and were selected to analyze the effects of changing the complexity of the task (i.e., considering or not background based task), while using App Inventor or MicroApp.

4.2 Hypotheses

Since the controlled experiment aimed at confirming that on average the use of the MicroApp environment both reduces the generation effort and simplifies the generation procedure with respect to the use of App Inventor, the following null hypothesis $H_{01}$ and $H_{02}$ have been formulated:

- $H_{01}$: the use of the MicroApp environment does not significantly affect the effort (measured in terms of the task accomplished time) to generate a mobile application;
- $H_{02}$: the use of the MicroApp environment does not significantly simplify the generation of a mobile application (measured in terms of the number of error and undo operations).

The alternative hypotheses are:

- $H_{a1}$: the use of the MicroApp environment significantly affects the effort to generate a mobile application;
- $H_{a2}$: the use of the MicroApp environment significantly simplifies the generation of a mobile application.

4.3 Selected variables and experiment design

In order to properly design the experiment and analyze the results, the following independent variables are considered:
• **Method:** this variable indicates the factor on which the study is focused, i.e. MA and AI (see Section 3.1).

• **Task:** the application generation tasks described in Section 4.1.

Two are the selected dependent variables: **Effort** and **Simplicity.** The former is measured as (i) the time (expressed in minutes) needed to accomplish the task. This was recorded directly by the supervisor noting down the start and stop time. On the other hand, Simplicity is computed by considering (ii) the number of errors reported by the tool during the editing and deploying phases, and (iii) the number of operations performed by the user during the editing of the application.

The considered treatments are all combinations of the factors Method (MA and AI) and Task (T1 and T2). To avoid results to be biased by group ability, each user experienced both Methods and both Tasks over the two subsequent laboratory sessions Lab1 and Lab2. Also, to minimize the learning effect, we needed to have users starting to work in Lab1 both with and without MA on both the generation tasks. Table 1 summarizes the design of the experiments, where $T_iM_j$ indicates the combination of task and method performed by a group of users in each laboratory session.

The subjects were assigned to each treatment in Table 1. We assigned the same number of subjects (i.e., 6) to each treatment, thus balancing the design. The groups were composed considering the cumulative score reached by the participants in the pre-experiment evaluation, in such a way to get a homogenous composition.

### Table 1. Experiment design

<table>
<thead>
<tr>
<th>Groups</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lab1</td>
<td>T1 MA</td>
<td>T1 AI</td>
<td>T2 MA</td>
<td>T2 AI</td>
</tr>
<tr>
<td>Lab2</td>
<td>T2 AI</td>
<td>T2 MA</td>
<td>T1 AI</td>
<td>T1 MA</td>
</tr>
</tbody>
</table>

#### 4.4 Preparation

The study has been divided in three steps and performed in one-to-one session (i.e., a supervisor for each subject) using the think aloud technique. In the first step, a lesson of 20 minutes introduced to all the subjects the principles of editing MicroApps and the main features of the prototype. Other 20 minutes were devoted to an introduction of the usage of App Inventor. To give subjects more confidence with the two tools, some examples (not related to the tasks to avoid biasing the experiment) were also presented to the subjects. The training sessions of the controlled experiment were concluded presenting detailed instructions on the tasks. To collect information concerning the perception of the experience of the subjects, we adopted the subjective usability approach proposed by IBM in [10]. In particular, the subjective evaluation has been collected considering two questionnaires:

- the Post-Study System Usability Questionnaire (PSSUQ) [10], consisting in 19 questions evaluating user overall system usability (OVERALL) focused on 3 subscales: System Usefulness (SYS USE), Information Quality (INFOQUAL) and Interface Quality (INTERQUAL). Also in this case a 7-point Likert scale anchored at 1 by Strongly Agree and at 7 by Strongly Disagree has been adopted, but each answer contains an open “Comment” space to collect deeper details about user impressions
- the Additional Questionnaire (AQ), shown in Table 2, where Q1-Q4 are proposed to assess the clarity of the tasks, while Q24-Q28 aim at better investigating the visual editor features and the features of the generated application. Also in this case a 7-point Likert scale anchored at 1 by Strongly Agree and at 7 by Strongly Disagree has been adopted.

The PSSUQ questionnaire should be proposed at the end of the evaluation to assess the usability of the considered tool. In our case, we evaluate a different tool in each task. Thus, each participant filled in both the AQ and the PSSUQ questionnaires at the end of each laboratory session.

Regarding the preparation of the devices involved in the controlled experiments, we installed a prototype supporting the composition of MicroApps on an Android based Samsung Galaxy S device, SDK version 2.3.3, to support the tasks performed using the MA method, while the App Inventor editor has been installed on a LAN Internet connected PC.

During the experiment, the supervisor did not provided any help to the subjects to avoid biasing the experiment. He only wrote the comments and problems of the subjects, when they spoke aloud. For each subject the needed time, the errors reported by the tool and the editing operations to accomplish the experiment was annotated as well. A camera shoots all the user operations.

#### 4.5 Material and execution

The participants accomplished each laboratory session without time limit. All the experiment material is available at the experiment web page\(^4\).

### Table 2. Additional questions of the Questionnaire-based survey

<table>
<thead>
<tr>
<th>ID</th>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>The activity objectives were clear to me</td>
</tr>
<tr>
<td>Q2</td>
<td>The activity to perform was clear enough to me</td>
</tr>
<tr>
<td>Q3</td>
<td>The material I received was enough to perform the activity</td>
</tr>
<tr>
<td>Q4</td>
<td>The task to perform was easy</td>
</tr>
<tr>
<td>Q24</td>
<td>The application composition rules offered by the tool editor are easy</td>
</tr>
<tr>
<td>Q25</td>
<td>The number of steps to perform an application is appropriate</td>
</tr>
<tr>
<td>Q26</td>
<td>Icon names and objects have a clear meaning</td>
</tr>
<tr>
<td>Q27</td>
<td>Each set of operations produce a predictable result</td>
</tr>
<tr>
<td>Q28</td>
<td>The application you generate matches your expectations</td>
</tr>
</tbody>
</table>

\(^4\)http://www.dmi.unisa.it/people/risi/www/MicroAppUsability
5. RESULTS

In this section, we present the results of the experiment survey questionnaire and draw some conclusions with respect to the empirical analysis.

5.1 Survey results

All the participants completed the tasks. Figure 5 summarizes the data concerning the task clarity considering the first four questions of the survey questionnaire, corresponding to Task 1, factor MA and AI, Figure 5(a) and (b), respectively, and Task 2, factors MA and AI, Figure 5(c) and (d), respectively. In particular, the objective was clear for both tasks and methods (question Q1). The activity to be performed (question Q2) was perceived as a bit clearer in MA for Task 1. The provided material was positively evaluated in all cases (question Q3), while some additional difficulties has been perceived during AI sessions with respect to MA ones (question Q4) for Task1. This perception increases together with the task complexity, as shown in Figure 5(c) and (d).

![Figure 5. Task clarity results](image)

Figure 6 reports the PSSUQ results aggregated in three categories, considering Task1 and Task 2 on both methods MA and AI: SYS USE, whose score is the average of the responses Q5-Q12, INFOQUAL that takes into account the questions Q13-Q19, and INTERQUAL, based on the responses Q20-Q22. Moreover, we also consider the OVERALL category, which summarizes the average of the responses Q5-Q23.

The participants diffusely perceived both the system as useful for Task 1, even if MA reaches a better consensus. This result worsens for Task 2 (the median of SYS USE was 3 for MA and 4 for AI). Concerning the quality of the information provided by the systems (INFOQUAL), analyzing the single question scores, it was evident that the participants had different opinions on the error messages useful to fix problems (PSSUQ Q13, $\mu_{AI} = 3.4, \sigma_{AI} = 1.17$ and $\mu_{MA} = 2.7, \sigma_{MA} = 1.35$), while they consider easier to find the information they need in MA modality (PSSUQ Q16 $\mu_{AI} = 2.3, \sigma_{AI} = 0.86$ and $\mu_{MA} = 2, \sigma_{MA} = 1.02$).

A specific question group of PSSUQ questionnaire is devoted to evaluate the interface quality (INTERQUAL). Also in this case MA is better perceived for Task2. In particular, two specific questions required the users to score how the interface is pleasant (PSSUQ Q20), $\mu_{AI} = 2.5, \sigma_{AI} = 1.2$ and $\mu_{MA} = 1.87, \sigma_{MA} = 0.68$, and how much they liked it (Q21), $\mu_{AI} = 3.8, \sigma_{AI} = 1.68$ and $\mu_{MA} = 2.1, \sigma_{MA} = 1.25$. Even if a bit less pleasant, users like more MA interface.

![Figure 6. The PSSUQ questionnaire results](image)

The OVERALL usability factor provides the overall system rating and is defined by the full set of items of the three subfactors. As shown in Figure 6, better results are reached for Task 1 MA ($\mu_{MA} = 2.6, \sigma_{MA} = 0.61; \mu_{AI} = 2, \sigma_{AI} = 0.50$) because the upper quartile is greater than the median. Referring to Task 2, the overall satisfaction decreases in both methods, but MA is always better judged. The last question of the PSSUQ questionnaire (i.e., "Overall, I am satisfied with this system") again follows the trend of its factor. Indeed, we have for Task1 $\mu_{AI} = 2.67, \sigma_{AI} = 0.89; \mu_{MA} = 1.83, \sigma_{MA} = 0.72$, and for Task 2. ($\mu_{AI} = 3.58, \sigma_{AI} = 1.08; \mu_{MA} = 2.5, \sigma_{MA} = 0.52$).

We also evaluate the perception of the different categories of users. In particular, we considered three categories Programmers, Intermediate and Non-Programmers, classified considering the pre-experiment questionnaire (PRO $\leq 3, 3 < PRO \leq 5, PRO > 5$, respectively). As shown in Figure 7, the opinions of programmers on MA are not homogeneous, but they like it more than AI. Some programmers preferred App Inventor, as the supervisor revealed, because they liked the block editor. Also Non-Programmers visibly preferred MA. Intermediate had a positive and uniform opinion on MA and a non homogeneous opinion on AI.

The user answers to the additional questions Q24-Q28 are resumed in Figure 8. In particular, let us note that the composition rules offered by MA were considered easier (question Q24), especially for Task 2 (Figure 8 (c) and (d)). As a deeper analysis revealed, non programmer users perceives App Inventor more complex. Also the number of steps needed to perform the task is more appropriate in case of MA and specifically for Task2 (question Q25). The same considerations hold for the icon names (question Q26). A participant commented, in the open section of the MicroApp evaluation: "The icon names are too cryptic".

![Figure 7. Comparison among the different categories](image)
Concerning the predictability of the operations (question Q27), there is a little preference for App Inventor. The application generated by App Inventor gets best results for Task 2 (Q28). Indeed, the MA interface is automatically generated and less appealing.

5.2 Empirical analysis results
For the empirical analysis we have considered three variables: (i) the number of minutes required to perform each task, (ii) the number of errors reported by the tool, (iii) the number of undo operations performed by the user during the editing of the application. We summarize the results in the BoxPlots shown in Figure 9, where errors and the undo operations concerning the two factors MA and AI are shown for Task 1 (Figure 9(a) and (b)) and Task2 (Figure 9(c) and (d)), while the results related to the accomplished time are shown in Figure 9(e). Let us note that the number of errors for MA and AI is quite similar for Task1, while the number of undo operations grows for the AI method in case of Task2. Indeed, the median of T2_AI undo is 12, while for T2_MA is 7.

The p-values for the Shapiro-Wilk normality test, executed on MA and AI, scored more than 0.05 in all cases except for T1_AI error distribution (p-value = 0.046), T1_MA time distribution (p-value = 0.013), and T2_MA error distribution (p-value = 0.009). These test results let us reject the hypothesis of a normal distribution for scores and required the adoption of a non-parametric test in evaluating the hypothesis. The Wilcoxon rank sum test revealed that, concerning the accomplishment time, the null hypothesis $H_{11}$ can be rejected (p-value = 0.034) and (p-value = 0.003) for Task1 and Task 2, respectively. Thus, we can affirm that there is a significant difference in productivity when using the MicroApp environment. Concerning the number of undo operations, p-value = 0.044 for T1, and p-value = 0.003 for T2. Thus, in case we consider only the number of undo operations, MA is simpler than AI. On the contrary, the number of errors for T1 has p-value = 0.63, and, consequently, no significance difference, while for T2 p-value = 0.033. Thus, the results concerning the simplicity in terms of errors and undo operations together provide no significance difference for the number of errors in case of Task 1 (p-value = 0.63), while the difference exists for Task 2 (p-value = 0.03).

If we group together errors and undo operations for Task1, we cannot reject the null hypothesis $H_{12}$. Thus, we can observe that the difference between MA and AI in terms of simplicity becomes significant with complex tasks.

5.3 Threats to validity
To evaluate the validity of the results we discuss the threats that may impact our experiment.

Concerning the internal validity threats, investigating eventual influences that can affect the independent variables with respect to causality, they are mitigated by the experiment design. Indeed, mono-operation and mono-method biases are avoided thanks to the adoption of two tasks and two methods. In addition, all the subjects were volunteers and, as consequence, motivated. Concerning the risks due to history and maturation, the circumstances are the same in both the lab sessions and the two
sessions occurred on the same day. The effect provided by the experiment instrumentation has been resumed in Figure 5 (Q4), where it is shown that the perception of the material provided was positive. As a consequence, this factor did not negatively affect the experiment. The construct validity of the questionnaire is assured by the adoption of a standard questionnaire [10]. However, the measurement of the variable was performed considering the times and undo operations.

External validity describes the study representativeness and the ability to generalize the results outside the scope of the study. We identified the following threats to external validity. The subject population we selected was not composed only of programmers, but of people having a mix of skills, as shown in Figure 4. Concerning the setting, the assignments consist in the development of small functionalities, which is the application target we aim at generating by using the MicroApp tool.

Regarding the conclusion validity, we consider both subjective and objective measurements. The adoption of a standard questionnaire excludes threat such as poor wording.

6. CONCLUSION

In this paper we performed an empirical evaluation aiming at assessing the effectiveness, the efficacy and the user satisfaction when generating a mobile application using the MicroApp tool, running on an Android mobile device. The results of this investigation provided evidence that, even if the mobile interface is restricted in size, the MicroApp tool is well judged, when compared with a similar generation tool, App Inventor, proposed by Google and runnable on a PC. Indeed, both objective measures, such as accomplishment time of tasks and undo operations, and subjective measurements, provided by a survey questionnaire, reveal very satisfying results for Micro App and for the adopted visual metaphors, especially when a task is not too simple. The evaluation revealed that positive results are reached, in particular, in case of non programmer users. This is relevant, because of the disadvantages of mobile devices, such as limited screen size. It is important to point out that the paper does not perform a full comparison of MicroApp and App Inventor. Indeed, App Inventor has a broader range of capabilities of the ones considered in the evaluation tasks, but the results of the evaluation are still interesting in that MicroApp appears to be effective for the level of tasks.

Future work will be devoted to engineer the prototype, adding the support information that is incomplete in the present version and improving the icons name and readability. We also plan to replicate this study adopting the engineered version of the tool with subjects with different backgrounds.

REFERENCES