Understanding End-User Development of Context-Dependent Applications in Smartphones

Gabriella Lucci, Fabio Paternò

Understanding End-User Development of Context-dependent Applications in Smartphones

Gabriella Lucci & Fabio Paternò

CNR-ISTI, HIIS Laboratory
Via Moruzzi 1, 56124 Pisa, Italy
{gabriella.lucci, fabio.paterno}@isti.cnr.it

Abstract. We are using our mobile devices in an increasing number of dynamic contexts, thus we need more and more context-dependent applications. However, only end users can know the most appropriate ways their applications should react to contextual events. In order to facilitate end user development of context-dependent applications in smartphones a first generation of mobile environments has been proposed in the market. In this work we analyse three such Android applications in terms of their ability to express the relevant concepts and their usability, also through a user study. We indicate some limitations of the current solutions and provide indications that can support future work for providing more effective results.

Keywords: End-User Development. Context-dependent Applications, Smartphones

1 Introduction

In recent years we have witnessed the use of computers in an increasing number of dynamic contexts. The number and variety of users of computational devices and tasks are increasing [1]. Users' backgrounds can vary from management, engineering, construction, education, research, health, insurance, sales, administration or other areas. However, such users share a common requirement for software to support their common tasks, which may vary rapidly. With a different range of backgrounds, their software needs are diverse, complex, and require frequent modifications. On the other hand, slow software development cycles and the lack of domain knowledge on the part of software developers are limitations to addressing the requirements of different users. End-user development can help to mitigate this gap.

Lieberman et al. [2] defined End-User Development (EUD) as "a set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify or extend a software artefact". The main EUD approaches have mainly considered the desktop platform and applications that are unable to adapt to the changing context of use. For example, desktop spreadsheets have been the most used EUD tools so far. Often, EUD approaches support users in composing and customizing sets of available basic elements

developed by programmers. Such basic elements are represented by and composed through intuitive metaphors, such as the jigsaw in which the basic elements correspond to the pieces to compose, or iconic data flow representations in which the icons correspond to the basic elements. HANDS [3] was an interesting contribution in investigating how to use HCI techniques to design a more usable programming system. It was a desktop environment exploiting an event-based language; the events considered were those related to application objects, while in this study we focus on how to indicate the reaction to events occurring in the context of use.

More recently, some EUD work has considered user mobility but in a limited manner. MIT App Inventor [4] is an EUD tool that allows users to create applications on the desktop to be executed on Android mobile devices, thus it still does not support development directly on the mobile device. MIT App Inventor expresses the application logic through OpenBlocks [5], where programming is performed by combining jigsaw pieces. Domain specific tools have also been developed targeting context-sensitive applications ranging from support for a set of template applications for tourism [6], domain-related content management to support guided tours [7], up to an EUD environment using concepts such as event-based rules and workflow rules [8].

More generally, desktop EUD environments lack the advantages of enabling end-users to create applications opportunistically in mobile scenarios. Recent advances in smartphones have enabled the creation of mobile EUD environments. Contributions for mobile EUD address: easy parameter contextualization for mobile applications, as in Tasker [9] (one of the Android apps considered in the study presented in the following); frameworks to support mobile authoring and execution, as in Puzzle [10]; creation of UIs through sketching or by adding interactive techniques in the touch screen [11].

The evolution of EUD for mobile applications is particularly important because users' mobility and the usage of smaller screens drive the need for context-awareness. However, the research in this field is currently at an early stage. Floch [12] describes the initial design of a city guide that can be tailored by end-users in order to include information from different service providers according to the visitor's position and visiting purpose. Our aim is to reach a better understanding of the requirements that should be satisfied by general end-users development environments that also offer the possibility to specify how the interactive application should behave according to the context of use. A first contribution in this direction has been recently put forward [13] but it has only considered practical trigger-action programming in the smart home by using the IFTTT ("If This Then That") Web environment, which still requires some technical knowledge greater than that of average persons.

More precisely, in the study that we present, we have considered three Android apps, which aim to support even user without programming knowledge to define their context-dependent applications that exploit the smartphones' sensors and capabilities. We have conducted an analysis of the three environments from two viewpoints: expressiveness (to what extent they support the relevant concepts) and usability (for which a user study has been carried out). We conclude with some indications drawn from this study that can inform future work.

2 The Considered EUD Environments for Smartphones

We focused our study on smartphone environments that allow non-professional developers to create context-dependent applications. For this purpose we found three Android Apps: Tasker, Locale[1], and Atooma[2] that provide different solutions. In this section we briefly describe each of them in order to better understand the analysis reported in next sections. It is worth noting that they provide three different solutions for supporting specifications of context-dependent applications according to the event / condition / action model. In general terms, an *event* is something that happens at a given time. Elementary events occurring in the interactive application or in the context of use, or a composition of such events. A *condition* is a specific constraint that should be satisfied, it can refer to something that happened before or some state condition. An *action* is the description of how the interactive application should change in order to perform the requested adaptation. In Section 3 we provide a more detailed analysis of the expressiveness of each environment.

2.1 Tasker

Tasker is derived from the evolution of an application designed to customize the capabilities of Palm OS handhelds available since 2007 and known as APT. The development of Tasker started in 2009, and then was expanded to extend the functionality of APT, and make it compatible with Android devices. At the end of the same year Tasker was awarded third place in the Android Developer Challenge 2. Tasker is available at a cost of 2.99 Euros and currently the number of its downloads is in the range 100000-500000.

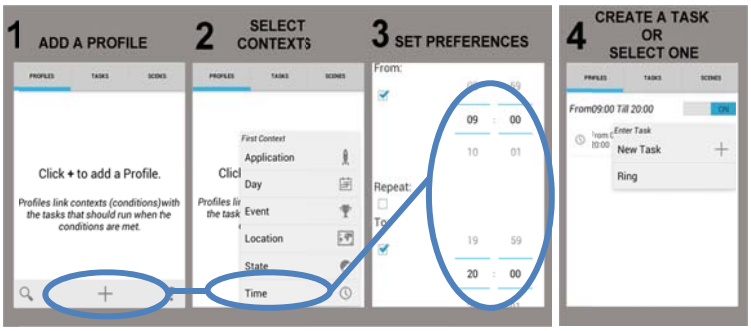


Fig. 1. Example of Tasker User Interface

In Tasker an application is called a profile. In order to develop an application the user can freely choose to start with the definition of either the triggering condition (which

[1] http://www.twofortyfouram.com

[2] https://play.google.com/store/apps/details?id=com.atooma&hl=it

is called context) or the actions (grouped in tasks). Unlike other solutions, in Tasker when a context type is selected for specifying a condition, then it is no longer available for another use within the current application (profile). A Tasker profile can contain up to four conditions. In addition, Tasker provides some specific functionalities to better handle the elements. For example, it supports an invert functionality that changes the state of an element (e.g. wifi active with this function becomes wifi deactivated). Figure 1 shows the initial steps in creating a context-dependent application in Tasker. The user starts a profile, and indicates first the type of context (Time in the example) and then specifies the corresponding parameters (in this case from 9:00 to 20:00).

Tasker requires that when users specify a condition then they have to immediately indicate one or more corresponding actions. If an additional condition is to be added, then at the end of the profile creation the condition specified firstly should be selected, followed by the Add button in order to indicate the additional condition.

As shown in Figure 1, at first the Tasks and Scenes tabs are also available. Tasks allows users to identify a set of actions that can be associated to a condition later on, while Scenes allows users to personalize some user interface elements that can be used to trigger a task. Tasker also allows users to specify what should happen when a condition is no longer satisfied. A further feature is the use of variables that allows users to indicate actions for unknown data in advance, for example to visualize in a notification the current state of a connection or to send an SMS when a call is missed.

2.2 Atooma

Atooma stems from the work of a young startup. Its first appearance was in September 2012, when the application became available on the Android Market and can be downloaded free. At the Mobile World Congress in 2013, Atooma won the Mobile Premier Award competition, obtaining first place among more than a thousand concurrent applications. To date, the number of downloads of the app is between 100,000 and 500,000.

Atooma represents the condition/action model using the IF … DO … representation through circular representations to indicate the relevant categories and elements. Figure 2 shows how to specify with Atooma whether the battery level is less than 20%. It requires four steps, one to select the category of the condition (mobile), next the type (battery), then the attribute (level), and lastly the corresponding value (less than 20%)

Fig. 2. Example of Atooma User Interface

In Atooma an application can contain up to five conditions (even of the same type) and up to five actions. It is also possible to share the Atooma applications with other users.

2.3 Locale

Unlikely Atooma, Locale is not a free application but can be purchased for 7 euro and 99 cents. Its release took place in 2008, and the number of installations is in the range of 50,000 / 100,000.
The main feature is its extensibility. The application in fact has a limited number of integrated elements but allows users to extend its functionality through the use of more than 400 Plugins downloaded from the Android Market (free or pay-for). The process of creating a Situation is not sequential: the user can arbitrarily decide the selection order of the elements because the interface is designed to show simultaneously (i.e. on the same screen) buttons for adding conditions and actions (in this environment they are called Settings), as shown in Figure 3.

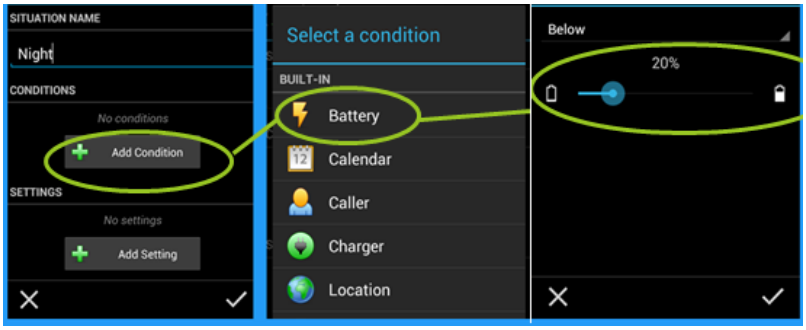


Fig. 3. Example of Locale User Interface

In Locale there is no maximum number of conditions to include in a Situation. It is also possible to specify actions to perform when no condition is satisfied.

2.4 Android Applications: A First Comparison

Even if we have introduced three applications for the same platform (Android Smartphones) with similar goals, there are some differences in the way they support users in achieving them. In general, they use slightly different vocabularies for the same concepts: application, event, action. In Atooma an application is called Atooma and is structured in an IF and a DO part. Locale supports the development of situations described in terms of Conditions and Settings. Tasker is used to create profiles structured into Contexts and associated Actions. In terms of number of events and actions to specify, Atooma limits them to a maximum of five for both; Tasker only limits the events (max four), while Locale does not specify any limit.

In terms of the development process, Atooma is completely sequential: developers have first to indicate the conditions and then the actions. In Locale it is not sequential and developers can freely choose to specify situations and settings in any order. Tasker is semi-sequential in the sense that the starting point can be either the condition or the action, but if a condition is specified then the corresponding actions must be indicated. Atooma and Tasker also support the sharing of the small context-dependent applications created. Tasker also allows the specification of exit tasks, which are actions to perform when the condition associated with the current rule is no longer verified. In addition, Tasker also supports the possibility to specify an execution order among rules that are triggered at the same time.
The support of logical operators in the conditions definition is rather limited: Tasker supports the specification of the NOT operator through the INVERT element (e.g. it is possible to specify conditions such as “the Bluetooth is not connected”), while Locale allows users to indicate OR conditions.

3 Expressiveness

In order to analyse the expressiveness of the environments in terms of their ability to support users to specify the relevant concepts, we have focused on the triggers that they allow users to indicate and the corresponding effects. Indeed, the three environments differ in terms of how they model what can be specified (events and actions). For example, right at the beginning Atooma asks users to select from four main macrocategories, Locale provides a list of conditions with some possible parameters, which can be extended through plugins, while Tasker structures the selectable events and conditions in terms of six Contexts. If we gather the elements that all three environments provide for both parts we can obtain a structure as indicated in Figure 4. Note that what can be specified in terms of conditions is similar to what can be specified in terms of actions, but there are also some differences: conditions can also depend on locations, while actions can also generate alerts.

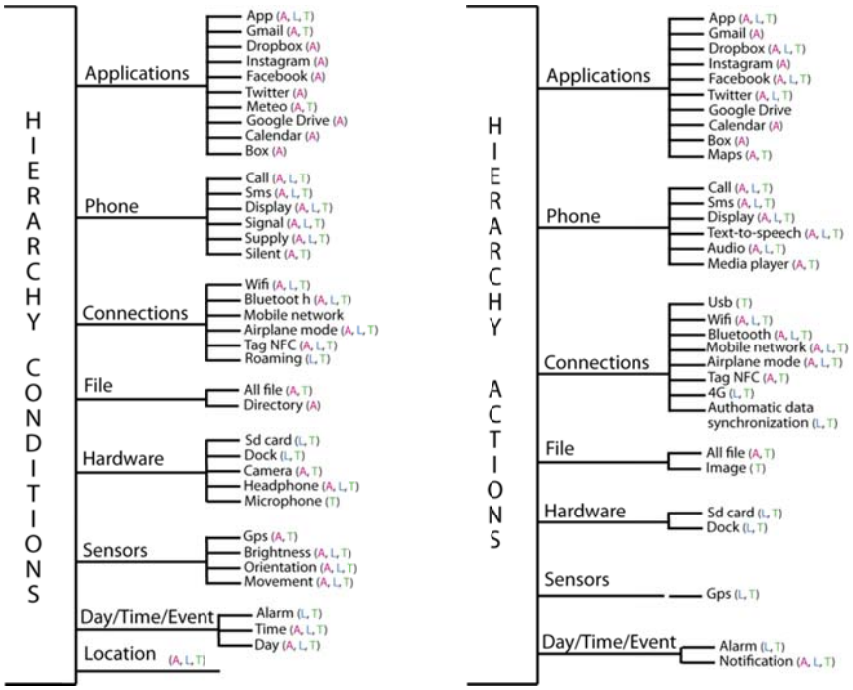


Fig. 4. List of all elements managed by the three environments collectively

Figure 4 shows which application supports each element by adding their initials (A for Atooma, L for Locale, and T for Tasker). It is possible to note that the three applications support only partially overlapping sets of elements. Figure 5 indicates how many triggers and actions can be specified through each environment. While Locale supports the two aspects in a balanced manner, Atooma supports more events than actions, and Tasker more action than events.

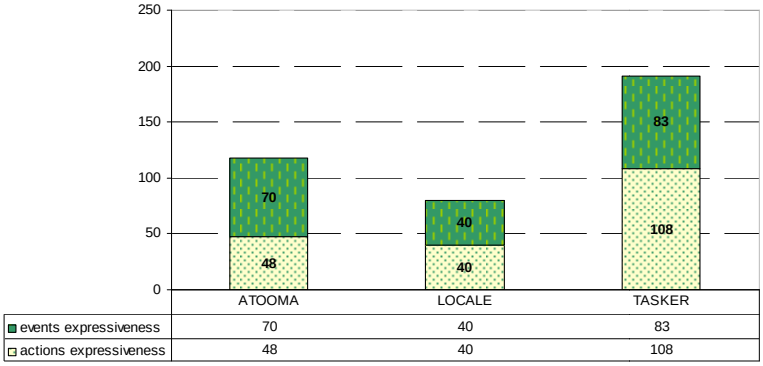
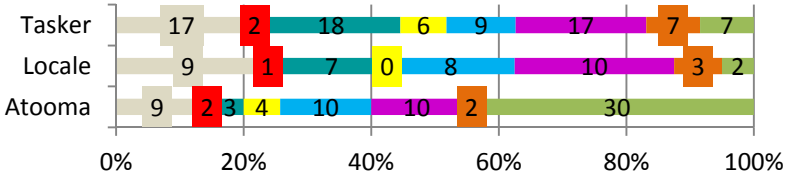


Fig. 5. Quantitative comparison of the concepts that can be expressed in the three environments

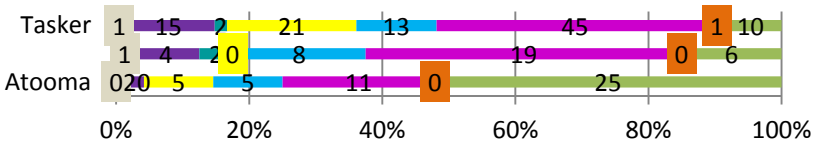
We can also analyse more in detail what they support with the following diagrams, one for the triggers and one for the actions. Tasker has the greatest expressiveness (more than double Locale's) and a number of actions that can be expressed (108) greater than the triggers (83). These numbers are to be attributed to the abundance of support in almost all categories. Figure 6 and 7 show that Tasker has a number of features always greater than or equal to Atooma and Locale, with the exception of the Applications (triggers and actions) and Connections (triggers) categories.



	Atooma	Locale	Tasker
Sensors	9	9	17
Location	2	1	2
Hardware	3	7	18
File	4	0	6
Connection	10	8	9
Phone	10	10	17
Day/Time/Event	2	3	7
Applications	30	2	7

Fig. 6. Comparison of triggers (event / conditions) supported by type

In Atooma the number of expressible conditions (70) is greater than the actions (48). Its expressiveness result is influenced mainly by the large number of features in the Applications category, but also by those of the Connections, Telephone and Sensors categories. The Applications and Telephone categories appear to be the most significant also as regards the functionality of the actions. In general, the ability to directly access the status and information of some of the applications installed on the device (including social networking applications, such as Facebook and Twitter) significantly increases the total value of its expressiveness.



	Atooma	Locale	Tasker
Sensors	0	1	1
Alerts	2	4	15
Hardware	0	2	2
File	5	0	21
Connection	5	8	13
Phone	11	19	45
Day/Time/Event	0	0	1
Applications	25	6	10

Fig. 7. Comparison of actions supported by type

In both triggers and actions, Locale results to have the same number of expressible functionalities (40) and, of the three environments, is the one that has the lowest total expressiveness. On a total of 80 features, 58 are obtained through plugins and this makes it even clearer how few elements are directly integrated into the environment. The presence of plugins, especially numerous in the Phone category, but also in the Sensors and Connections categories, allows the environment to significantly increase its degree of expressiveness, severely limited by the low number of features found in the Applications category and the inability to set conditions that relate to the File category.

4 Usability

In order to analyse the usability of the three environments we have carried out a user test with 18 users with some familiarity in the use of interactive applications on mobile devices though without any expertise in the field of computer programming. Each participant was asked to perform two tasks of increasing difficulty on all three Android apps. To avoid that users being influenced by the order in which they used the applications, the 18 participants were divided into six groups; each one associated with one possible combination in the order of using the three applications.
Before starting the test, each user filled out an anonymous questionnaire aimed at understanding the level of familiarity with mobile devices and their personal attitudes. As we shall see in the following sections, for each of the environment considered, the test was divided into three consecutive parts: a first training phase, a subsequent phase

of tasks execution, carried out by using the think aloud technique and recorded in audiovisual format, and a questionnaire. Once finished, the users' general impressions have been collected in a final questionnaire in which it was required to make judgments on a 1 to 5 scale regarding the three overall applications tested, and provide any comments.
The final stage of the evaluation was carried out by analyzing the quantitative data (number of users who completed the tasks, performance time, etc. ..) and qualitative information (comments, suggestions, etc. ..) contained in records and collected through questionnaires.
The versions of the applications used in the test were: version 1.2.6 for Atooma, version 5.0 for Locale, and version 4.2 for Tasker.

4.1 Training

Before running the test, a short learning phase was necessary in order to allow the users to understand the purpose of the test and the general functioning of the three Android environments. The learning phase was composed of three steps:

1. Brief verbal explanation of the conditions / actions model implemented by the context sensitive applications.
2. A video tutorial that shows the creation of the following mini-application example:

   - From 8 to 18 hours (condition)
   - If the battery level falls below 20% (event)
   - Show me a notification with a message such as "low battery" (action)

3. Two minutes of free use of the application in question to become familiar with its interface.

4.2 Tasks

Once they finished the learning phase, the users were asked to perform two tasks consecutively using the interfaces just introduced. As mentioned before, the order in which the environments have been proposed to the users depended on the group to which they were assigned in order to balance the learning effect. The two tasks were always accomplished in ascending order of difficulty for each environment. The first was to specify an application composed of one event and one action, the second application included an event, a condition and two actions, in particular they were:

1. When I'm home (condition) -> Activate Bluetooth (action)

2 When I insert earphones (event), if the device is oriented vertically (condition) -> launch the application " Tuneln Radio" - installed in the device (Action 1) and decrease the brightness of the screen (Action 2).

The users were asked to think aloud during the task performance, and their interactions with the three interfaces were recorded by using the Android app SCR Screen Recorder in order to collect further data (quantitative and qualitative) on which to perform the evaluation. The quantitative data we obtained include:

- the average time to perform the task
- the number of users who performed the task correctly
- the number of users who failed the task
- the number of suggestions offered to complete the tasks.

A person familiar with the environments was always present during the test sessions. Such person also provided suggestions, when requested. The suggestions were divided according to the type of difficulty encountered:

- serious misunderstanding of the trigger / actions model or complete misinterpretation of the application interface (in terms of the correspondence between conditions and actions and their logical structure, and how to obtain it in the environment). Such suggestions were associated with unsuccessful task performance because without them the users would not have been able to complete their tasks.
- medium, they were linked to poor interpretation of items and categories (they needed suggestions related how to understand the contents of the possible contexts and categories).
- negligible, if related to language difficulties (English translation of terms unknown to the user).

4.3 Users

The participants were 18 (8 females), their age was between 19 and 35 (average was 26). The majority of them (15 out of 18) possessed an Android device and used various kinds of interactive applications more than 5 times per day.
The messaging application WhatsApp was found to be the most utilized; 16 out of 18 users said to use it on a daily basis. Facebook was in second place (12 users), followed by applications for browsing and e-mail, in particular Google browser (12 users) and GMail (8 users). The mobile applications used by the fewest participants were Youtube, Messenger, Instagram, weather applications, geolocalization and navigation applications such as FourSquare and Google Maps.

4.4 Task Performance

Considering both the first and the second task performance (which means a total of 36 tasks for each environment), the applications that yielded the greatest and the lowest number of successful task performance were respectively Atooma (32, or 88.8%), and Tasker (28, i.e. 77.8%). Locale is between these two values (30, 83.3 %).

The suggestions of medium entity, which derived from misunderstandings related to the interface (misinterpretations of the elements, the content of the categories, etc. ..), are significant for our purposes, as they allow us to understand which types of representations were unclear.

One participant who requested the medium suggestion was performing the second task proposed by the Tasker environment and was highly uncertain about the function of Contexts. With Tasker, creating a mini-application starts in fact from the choice of one of the six Contexts offered (Application, Location, Time, Day, State, Event). The difference between the roles of the State and Event contexts made it more difficult for the user to find the condition corresponding to the insertion of the earphones. He was therefore advised to focus on the type of condition to be set, abstracting the meaning and trying to identify which of the two contexts could contain the required functionality. After about a minute the user was able to understand that the element of interest was to be found in the State context, since the coupling of the earphones refers to the state of that particular hardware device.

As mentioned before, all serious misunderstanding suggestions were provided when the user did not understand the mechanism conditions / actions or, to be more precise, in cases where because of this misunderstanding users failed to grasp the overall functioning of the application interface and therefore entered an action in place of a condition or vice versa. Thus, without such help the users were not able to complete the task, so they were associated with task failures.

In some cases, even when the users were able to perform a task successfully, their navigation within the application interface followed incorrect paths, a problem which was then corrected autonomously during the course of the test.

Atooma

There were three unsuccessful performances of the first task with Atooma (one user received support to complete and two did not complete the task correctly). In an attempt to set the condition for localization (when at home), two users selected the GPS element, setting it to ON, rather than using the Location attribute. In the questionnaire the two users commented on the error stating that their device requires activation before you can indicate a GPS location.

As for the second task, the main mistake in navigation was related to the setting of the actions, and concerned finding the Radio application to open upon insertion of the earphones. Almost all the users looked for it within the macrocategory Apps, since the label gave them the idea of being able to access all the applications installed on the phone.

Locale

For the first task there was an error in which the user selected a wrong action (notification instead of Bluetooth). For the second task, five users selected the item Screen Orientation Up, thinking that it was equivalent to the condition "if the device is in a

vertical position." The error was to proceed immediately to the Orientation element, in an attempt to set the condition on the vertical position of the device.

Tasker

For the first task three users made the same mistake: they set the Bluetooth Voice attribute (instead of Bluetooth), after having found it in the Settings category (rather than in the NET category). Even users who were able to perform the task had some initial concerns about which was the category in which to find the Bluetooth element, and therefore explored most of the categories relying more on reading the labels of the elements rather than the criteria of the logical categorization.

4.5 Performance Times

In the calculation of the average performance times we have considered only the tasks that were performed successfully.

First Task

With Locale the mean time of the first task execution was about half than Atooma. We can therefore easily infer that, as regards the performance of the first task, Locale was found to be more intuitive and understandable. In Atooma and Tasker the number of task successes and failures is absolutely identical (15 and 3). Comparing the respective average times, however, the average time for the execution of the first task with Tasker (3'50") was almost a minute longer than Atooma (2'52"). The time difference derives essentially from the fact that Tasker has a much higher number of categories and items than Atooma. Users spent more time searching the attributes necessary to complete the task and, in some cases, the abundance of categories meant that they committed errors whose correction increased the execution time of the task. In particular, twelve users were not able to immediately identify the NET category, which included the Bluetooth element, but explored almost all the categories (especially the Phone and Settings categories).

While the minimum time in Atooma (1'27'') and Tasker (1'21'') is similar, the difference between their respective maximum time is around a minute (5'41'' vs 6'44''). Locale shows the most interesting results: it not only has a smaller number of errors and a lower average execution time (1'29") than the other applications, but also the minimum (0'42'') and maximum (3'24'') times are much lower than those obtained with the other environments. Locale results to have the smallest value (0'42'') also with regard to the standard deviation vs Atooma (1'17') and Tasker (1'34''). This means that the differences between the times taken by participants to successfully perform the first task was not very high and that users were able to complete the task using roughly the same amount of time. In contrast, the standard deviation of Tasker and Atooma are much higher than that of Locale, showing extreme variability in the times achieved by the users.

Second Task.
In the second task, the application that required less average execution time was once again Locale (3’35’’) followed by Atooma (4’00’’) and Tasker (5’14’’). The difference between the average time between the latter two even in this case is about one minute. The fact that Locale has obtained the lowest average times in both the first and the second task is due to the greater simplicity of its internal structure. The elements of conditions and actions are grouped into a well-defined, compact set of categories, each one with a few parameters, and this feature makes it easier to search for the relevant element.
Regarding the minimum and maximum times, those of Locale (2’29’’ and 5’34’’) appear once again to be lower than those of the other applications, but unlike the first task, in this case the difference between the minimum and maximum times of Atooma and Tasker is much wider: Tasker min time (3’15’’) is almost a minute higher than Atooma (2’44’’), while the maximum time (10’27’’) exceeds it by almost five minutes (5’45’’). The maximum time obtained with Tasker is a clear indication of how complex is to specify more structured adaptation rules through it, also for the abundance of categories for selecting an element.
The data related to the standard deviation of the times in the second task are in line with those achieved in the first task, but reveal additional information. The fact that the standard deviation of Locale is lower in the first than in the second task (0’53’’) suggests that its value increases with the number of conditions and actions to be set. Atooma, on the contrary, obtained a standard deviation lower in the second (1’01’’) as compared to the first task and this suggests that, although the number of conditions and actions to be selected was greater, the execution time depended mainly on the user's familiarity with the interface.
The difference between the standard deviation for Atooma and Tasker is greatly increased in the second task. Tasker proves once again to be the application with the most variation in execution times (1’43’’), mainly due to the time spent navigating between available contexts and categories.

4.6 Discussion

The analysis of the data collected from the test allows us to obtain useful information on which to base our conclusions on the design of the considered environments, and formulate hypotheses for improvements.
Atooma obtained the highest number of successful performances (88.8 %) and the sum of the average execution times of the two tasks was 6 minutes and 52 seconds. The two main mistakes in the first task involved the setting of the current location because users did not notice the presence of the Position element and they immediately headed to the GPS element. This oversight is mainly due to the users being accustomed to dealing with the localization settings of their Android devices requiring prior activation of the sensor.
Among the most frequent navigation errors there was a tendency to think that the opening of the Radio application could be done through the APPS category, rather

than the Mobile. However, exploration of the Atooma interface, which was for many users enjoyable and fun, meant that users were able to correct their errors independently and without too much difficulty.

Here are some of the considerations expressed by the users:
"*The logical mechanism is very simple* " ;
" *The graphics are intuitive and the dial reminds me of the old phones* ";
" *IF and DO are easy to understand, the display is clear, you are aware of what you are doing* " ;
" *Once you understand, the mechanism is simple to use and visually intuitive*" ;
" *There are not too many categories and they are well organized* ."

The criticisms to the interface were as follows :

"*The dial can be confusing because there are a lot of elements in the Mobile category, the dial hides some of them and it is difficult to understand that the hidden elements are more numerous than expected*" ;
and again: "*it is easy to use but the spinning dial hides elements and raises the doubt whether there are further elements in that category* ."
"*The only flaw is that the elements with icons are not in alphabetical order*";
" *The colours of the major categories are too bright and distracting*."

Of the 18 users, 50% said they had received a high level of satisfaction from the use of the application; 33.3% an average satisfaction level, and 16.6% a very high level of satisfaction. The icons and bright colours of the application were particularly appreciated.

Locale scored a good overall percentage of successes (83.3%) and to successfully complete the two tasks users employed an overall average of 5 minutes and 4 seconds. There was no need for serious suggestions. This indicates that the application interface appeared so simple to use that users did not need to ask for external aid. Indeed, the error committed by the users who failed the task (we refer to the second proposed task) was to think that Screen Up corresponded to a vertical position. An oversight which, as users explained, was derived from the fact that inside the Orientation category that was the only sensible choice. Locale, as evidenced by its level of expressiveness, has a limited number of features, especially if we consider only those integrated into the application and not provided by the appropriate plugin.

The comments were mostly positive about Locale :

" *There is a clearer distinction between conditions and settings* " ;
" *The button to add conditions and settings is on the same screen, there is the confirmation button and then the items are listed for easy identification* " ;
"*All the elements are in sight because you select from a list* " ;

"*It has an intuitive interface because the structure is simple and it is not possible to get lost* " ;
"*The elements are arranged in a list with colour icons whose level differs greatly*" ;

55.5% of users expressed a high degree of satisfaction; 16.6% an average level and 27.7% a very high level of satisfaction. Locale was positively assessed also for the size and contrast of the icons on the dark background of its interface, which were immediately visible. All 18 users stated that they would recommend it to anyone using the app for its extreme ease of use.

Tasker obtained a total of 77.8% of successful performance, the lowest percentage. The overall mean time to carry out the two tasks was 9 minutes and 4 seconds. There was a need for four main suggestions and three users carried out errors. Such errors were caused by a misinterpretation of the categories. In particular, the three users in question were highly uncertain about which category contained the Bluetooth element.
What created the most problems to the users was the high number of categories, with many elements within them often represented with the same icon; the presence of Contexts (States and Events) with doubtful meaning; and unclear limitations on how to specify sequence of events/conditions and actions. Indeed, in Tasker after indicating one event it was not possible to immediately specify further events but users had then to indicate the corresponding actions and only afterwards could add further events.

The positive comments on the Tasker interface were limited:

" *Not very simple and intuitive, however, it is highly functional and well categorized*";
" *Fairly intuitive* ."

Users provided more negative opinions :
"*The difficulty is not so much the functions but the whole program*" ;
" *Having to put the action immediately after the first condition makes it difficult to use* " ;
"*It was difficult because it is not clear that I have first to set a condition and after the actions; if I had a profile already created by another person I would not understand anything* " ;
" *The graphics are poor and do not facilitate the use of the app* " ;
" *There are many categories and the elements have the same icon as the parent category* " ;

Various users (38.8 %) said they had received an average satisfaction after testing the application; 22.2% , however, expressed a low level of satisfaction. 11.1 % said they were frustrated by the use of the app (level of satisfaction very poor), the same num-

ber, on the contrary, said they got a high degree of satisfaction . 16.6% gave a very high level of satisfaction.

Eight of the respondents (44.4%) also indicated that, because of the complexity of Tasker due to the huge amount of categories and items, they would not recommend the application to a friend. All others indicated that they would recommend it to friends only with a minimum of technological experience.

The last question was about which environment the user would use. 44.4% of users chose Locale, 33.3% of them Atooma and only 22.2%, attracted by the expressive potential of Tasker, chose it even at the expense of ease of use.

5 Conclusions and Future Work

This study has provided useful insights on the main features that an environment for end-user development of context-sensitive applications should have. One first issue is associated with the lack of consistent terminology, each environment provides different names for similar concepts, which does not help users to immediately understand them.

We have seen how the most expressive environment (Tasker) is also the one that was found most difficult to use (highest performance time, error numbers, and unsuccessful performance numbers). This provides some interesting indications of the conflict between expressiveness and usability. Thus, there is a need for novel solutions that are able to support high expressiveness as well as usability. It is clear that with the increasing number of categories for grouping the relevant concepts, there is also an increasing risk of misunderstandings unless familiar classifications, icons and metaphors are proposed to represent and manage such concepts. In addition, it is important to improve the user experience in this type of environment, since too much effort in learning and understanding the relevant concepts discourages its use.

The elements that can characterise a novel solution able to improve the user experience in authoring end user development of context dependent applications include: the choice of elements terms and icons immediately understandable without ambiguity; since there are many possible elements they should be structured according to intuitive logical categories that match the mental representation of mobile users; the ordering in specifying events, conditions, and actions should be flexible without artificial constraints; usability should not be at the expense of expressiveness, thus it should be important to still allow users to easily indicate flexible events, conditions and related actions in which the elements can be composed according to various logical and temporal operators, without any particular limitation on the number of events and actions to compose.

Future work will be dedicated to better understanding how users classify the concepts that characterise context-dependent applications using a card sorting technique,

and then to design a new environment for smartphone able to take into account user mental models and support the possibility of specifying context-dependent applications through an expressive language, such as the AAL-DL (Advanced Adaptation Logic Description Language) [14], in such a way to allow even end-users to exploit it.

References

1. C. Scaffidi, M. Shaw, B. Myers, (2005): Estimating the Numbers of End Users and End User Programmers. In: VL-HCC 2005 - IEEE Symposium on Visual Languages and Human-Centric Computing 21-24 September, 2005, Dallas, TX, USA. pp. 207-214
2. H. Lieberman, F. Paternò, M. Klann, V. Wulf, “End-user development: An emerging paradigm” In: Lieberman, Henry, Paternò, Fabio and Wulf, Volker (eds.). "End-user development (Human-Computer Interaction Series)", pp. 1-8, Springer, 2006.
3. J.Pane, B.Myers, L.Miller, Using HCI techniques to Design a More Usable Programming System. Proceedings IEEE HCC’02, pp.198-206.
4. http://appinventor.mit.edu
5. http://education.mit.edu/openblocks
6. G. Ghiani, F. Paternò, L. Spano, “Cicero Designer: An Environment for End-User Development of Multi-Device Museum Guides” In Volkmar Pipek, Mary B. Rosson, Boris Ruyter, and Volker Wulf, editors, End-User Development, volume 5435 of Lecture Notes in Computer Science, chapter 15, pages 265–274. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
7. A. Celentano, M. Maurizio, “An end-user oriented building pattern for interactive art guides” In M. Costabile, Y. Dittrich, G. Fischer, and A. Piccinno, editors, End-User Development, volume 6654 of Lecture Notes in Computer Science, pages 187–202. Springer Berlin / Heidelberg, 2011.
8. V. Realinho, T. Romão, A. Dias, “An event-driven workflow framework to develop context-aware mobile applications”. In Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia (MUM '12). ACM, New York, NY, USA, , Article 22 , 10 pages, 2012.
9. http://tasker.dinglisch.net
10. J. Danado, F. Paternò, “Puzzle: a visual-based environment for end-user development in touch-based mobile phones” In Proceedings of the 4th international conference on Human-Centered Software Engineering (HCSE'12), Marco Winckler, Peter Forbrig, and Regina Bernhaupt (Eds.). Springer-Verlag, Berlin, Heidelberg, 199-216, 2012.
11. J. Seifert, B. Pfleging, E. Bahamóndez, M. Hermes, E. Rukzio, and A. Schmidt. 2011. Mobidev: a tool for creating apps on mobile phones. In Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI '11). ACM, New York, NY, USA, 109-112.
12. J. Floch, “A Framework for User-Tailored City Exploration” In IS-EUD 2011, Proceedings of the Third International Symposium on End-user development, Torre Canne (BR) Italy, June 2011, Lecture Notes in Computer Science Volume 6654, pp 239-244, 2011.
13. B. Ur, E. McManus, M. P. Y. Ho, M. L. Littman: Practical trigger-action programming in the smart home. CHI 2014: 803-812
14. F. Paternò, C. Santoro, D. Spano, Serenoa EU Project, AAL-DL: Semantics, Syntaxes and Stylistics, 2013,