Analysing How Users Prefer to Model Contextual Event-Action Behaviours in Their Smartphones

Gabriella Lucci and Fabio Paternò(✉)

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

Abstract. Developing context-dependent applications involves indicating the relevant contextual events and the corresponding actions. Based on an analysis of the usability and expressiveness of three Android apps for developing such applications, we have started a study that aims to identify a general solution able to better represent how users classify the relevant concepts in order to facilitate their manipulation during development. We report on a card sorting experiment carried out with 18 users for this purpose, and an analysis of its results, with suggestions for improving current designs and informing future solutions.

Keywords: End-user development · Context-dependent applications · Smartphones

1 Introduction

The main End-User Development (EUD) approaches have focused on the desktop platform and applications that are unable to adapt to the changing context of use [1]. For example, desktop spreadsheets have been the most used EUD tools so far. Some environments allow the development of applications for mobile devices, but still through the desktop platform. One example is App Inventor[1], which provides a graphical environment for creating Android applications.

Only recently have some contributions also started to consider the smartphone as a platform in which the development can be carried out. In this respect, one important point to clarify is that the adoption of mobile devices does not only imply that the development platform has a screen with limited size with which to interact through touch. It also means that the corresponding applications have the potential to dynamically detect relevant information on the context of use through several sensors, and thus adapt their behaviour accordingly. Such contexts can vary in aspects related to the users (tasks, preferences, emotional state, …), the technology (devices, modalities supported, connectivity, …), environment (light, noise, place, …), and social aspects, and only end users can know the most appropriate ways their applications should react to contextual events.

[1] http://appinventor.mit.edu/explore/

P. Díaz et al. (Eds.): IS-EUD 2015, LNCS 9083, pp. 186–191, 2015.
DOI: 10.1007/978-3-319-18425-8_14

One possible solution for developing context-dependent applications, still in desktop platforms, even if the obtained applications can then be executed in mobile devices, is IVO (Integrated Virtual Operator) [2]. In IVO the authors used the workflow metaphor in which the activities are triggered by events that are automatically generated at runtime by sensing the environment, either through the smartphone's own sensors, or using a sensor infrastructure external to the smartphone.

More recently, some contributions aiming to support end user development on smartphones have been put forward. For example, Puzzle [3] proposes the adoption of the puzzle metaphor to support development of Internet of Things applications on smartphones. The supporting environment has been designed to facilitate the composition of various pieces through a touch interface for a screen with limited size. Thus, the tool provides a usable solution but limited to the composition of functionalities for which a puzzle piece has been provided.

An attempt to apply the programming-by-example paradigm to a mobile development environment is "Keep Doing It" [4]. It provides the possibility of identifying context-dependent adaptation rules in the event / condition / action format according to the history of user interactions. The rules are represented through a natural language subset using "when", "if" and imperatives verbs. Another environment that aims to support the development of small reactive applications is IFTTT. It uses the textual syntax "IF This Than That" to specify the scheduling of execution of a certain action (That), and the occurrence of a specified event (This). Its distinguishing feature is that, besides being able to express "recipes" that concern and make changes in the hosting device, IFTTT communicates with widely used Web services, thus allowing the automatic execution of functions related to the internal state of apps. A recent study [5] found that trigger-action programming can express the most often desired behaviours in order to customize smart home devices. They conducted a 226-participant usability test of trigger-action programming, finding that inexperienced users can quickly learn to create programs containing multiple triggers or actions obtained by extending the IFTTT language, which has limited possibilities, since it only supports applications with only one trigger and one action. This shows that this approach seems suitable to support EUD of context-dependent applications, but needs to be improved in order to allow users to express the various desired combinations of events and corresponding actions.

Our aim is to reach a better understanding of the users' mental models when they want to specify how their interactive applications should behave according to the context of use, and to provide design suggestions for EUD tools that better match such models.

In this paper we discuss a recent study that carried out a comparative assessment of three Android apps in terms of expressiveness and usability [6]. We then report on a follow-up card sorting study that aimed to better identify how users logically organise the concepts supported by such apps. Lastly, we provide a discussion aiming to analyse the current designs and provide suggestions for new designs that better match the requirements identified.

2 An Analysis of Android Apps for Context-Dependent EUD

An input for our work has been a recent study [6] on how three Android Apps (Tasker[2], Locale[3], and Atooma[4]) aim to support non-professional developers in creating context-dependent applications by exploiting the smartphones' sensors and capabilities. The analysis of the three environments has been carried out from two viewpoints: expressiveness (to what extent they support the relevant concepts); and usability (for which a user study has been carried out).

They provide three different solutions according to the event / condition / action model. They tend to structure the relevant concepts in similar ways in terms of categories, elements, actions. However, they use slightly different vocabularies for the same concepts. 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 Tasks composed of Actions.

Tasker has the greatest expressiveness (more than double Locale's), with a number of actions that can be expressed (108) greater than the triggers (83). In Atooma the number of expressible conditions (70) is greater than the actions (48). In both triggers and actions, Locale has the same number of expressible elements (40) and is the one that has the lowest total expressiveness. Of a total of 80 features, 58 are obtained through plugins since few elements are directly integrated into the environment.

The three environments differ in terms of how they model what can be specified (events and actions). Right at the beginning Atooma asks users to select mainly from four main macrocategories. Locale provides a list of elements, which can be extended through plugins, while Tasker structures the selectable events and conditions in terms of six Contexts.

We have noted a lack of consistent terminology in such Apps: each environment provides different names for similar concepts, which does not help users to immediately understand them. 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).

In general, 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. Since there are many possible elements to specify, 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. It can also be useful to 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. In addition, the set of events and conditions to consider should be extensible.

---

[2] http://tasker.dinglisch.net

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

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

3 Card Sorting for Identifying User Conceptual Models

In order to identify a more intuitive classification of the concepts that characterize context-dependent applications, we have carried out a user study through card sorting and associated cluster analysis techniques.

The identification of the cards used in the study derived from the analysis of the three Android apps mentioned in the previous section. We used cards that were associated with all the event and action types supported by the three apps. Thus, we obtained 39 cards: 14 referring to only events, 6 to only actions, and 19 were used for both events and actions.

The card sorting was proposed to 18 users with ages between 18 and 35 (average 27), 72.2% were males. In terms of the most used mobile operating systems, 66.9% used Android, 16.6% iOS, 11.1 Windows, 5.7 Blackberry. 72.2% use the mobile to access interactive applications more than 5 times per day.

At the beginning we provided them with some basic concepts to introduce context-dependent applications, then the users had to group them logically and assign a name to each group identified. We did not provide any particular constraint regarding the number of groups to create or limit the possibility of creating sub-groups. They had to carry out the exercise twice: once to classify the 33 cards related to events and once for the 25 cards representing the possible actions. In order to avoid any possible bias, half started with the events and half with the actions. During the exercise the groups identified by the users were entered in the UXsort tool[5], which has been used to support the results analysis. By applying hierarchical clustering methods the tool is able to measure the linkage among elements groups and produces a dendrogram that represents the similarity among elements through a tree-like structure. The tool supports their analysis by using three clustering algorithms (single linkage, complete linkage, average linkage).

In order to select the most interesting results, we decided to focus on solutions that yielded a number of groups between 5 and 8. Such numbers were identified by the analysis of the numbers of groups supported by the current solutions: Atooma groups the elements in five macrocategories, while Tasker exploits six contexts (two of which contain 10 and 12 elements). In the user test mentioned in the previous section the greater number of elements managed by Tasker implied higher expressivity, but also required the subjects to take longer to find the desired elements, especially when they were located in unexpected places in the proposed logical hierarchy. In addition, we considered that less than five groups results in some groups containing many elements, which can become confusing when users look for a specific item, and with more than 8 groups the solutions tend to separate elements that people would expect to be together. In the analysis we used such criteria to identify the levels at which to cut the dendrograms (there were three dendrograms: for single, complete, and average linkage) and obtain clusters with the required cardinality.

In the case of the events, we thus obtained eight possible solutions, which required further analysis: some were then discarded since the resulting groups contained an unbalanced number of elements (as far as having a group with two and one with eighteen elements), while others were discarded since the contained elements were

[5] http://www.uxsort.com/

rather heterogeneous (for example, in one case elements such as SMS-Call-Signal were grouped with elements such as Gmail-Facebook-etc.). Thus, we ended up with three similar solutions in which the only differences were related to the location of the elements Silent and the pair Airplane mode and Silent mode, which could be located together with either the group SMS-Call-Supply-Display or the group Bluetooth-WIFI-Tag NFC – Roaming – Mobile Network.

By observing the element patterns in the groupings that occurred in both the events and actions classifications, we have identified the associations between elements and groups that users found most meaningful, and the corresponding group names that were assigned most frequently. We also discarded solutions with an unbalanced number of elements in the resulting groups or with a group whose name was not completely consistent with the actual elements (e.g Archive for a group containing the Media player element). A similar process was followed to analyse how users grouped the 25 action types. Also in this case we discarded solutions with nine groups since they contained a widely varying number of elements (even with groups with only one element). Thus, we obtained twelve possible solutions that all contain six groups of elements that are always together: 1. [Dock – SD Card], 2. [File –Image], 3. [Airplane Mode – Automatic Data Synchronization – Tag NFC – WIFI – Bluetooth – Mobile Network], 4. [SMS – Call – Audio – Display], 5. [Alarm-Notification], 6. [Gmail-Facebook-Twitter-Instagram-App-Dropbox]. In this set of solutions we discarded some that included groups that would be expected to be autonomous together with other elements. Thus, we obtained five solutions that differ by three elements (Media Player, Text-to-Speech, GPS).

The classification of the nineteen elements in common between events and actions has been consistent in the resulting solutions. The following fifteen elements have been grouped in the same way: 1. [Dock – SD Card], 2. [Call – SMS – Display], 3. [Tag NFC – WIFI – Bluetooth – Mobile Network], 4. [Gmail – Facebook– Twitter – Instagram – App – Dropbox]. Airplane Mode, Alarm, File and GPS were the common elements that have been classified differently. The reason for such differences was that each of these elements was strictly related to specific elements in either the events category or in the actions category, thus when users had to classify them when the connected element was not available, then they had to look for new ways to classify them. For example, the GPS event was always grouped with [Location – Movement], but such elements were not available in the action category.

Users also provided suggestions regarding the names to associate with the logical groups. For some groups, users assigned clear preferences regarding their names (as in the case of App, Phone, Hardware). Connections and Sensors received a good number of preferences as well and were more general than other choices for their groups (such as Net and Localization). In one case there was a similar number of preferences for both Archive and File; the latter seems more appropriate. In the end, we found a solution with two small variants that differ regarding the location of the GPS element, which could be included in the groups “Connections”, “Sensors”, or “Localization”. In one case we obtained seven groups for both events and actions. The groups App, Phone, Hardware, File, Events, Connections (including GPS) in common for both, while for the events there was the Sensors group as well, and for actions there was the Multimedia group.

4 Design Implications

Of the elements considered in the card sorting exercise, 17 in the Atooma mobile category are classified differently from the results suggested by the users who preferred to distribute them across various groups. In contrast to Atooma, which supports the same classification for events and actions, in Tasker, while the events are grouped in six contexts (two of which with further subcategories), the actions are reachable through 20 categories, some of which still maintain the names of the event classification (Phone, Net, Display, App, …). Thus, the Tasker classification of the actions is rather different from that resulting in the card sorting exercise. Since the number of groups is higher in Tasker then some elements, which in the card sorting exercise are in the same clusters, are placed in different groups in Tasker. In general, Tasker supports a great number of operations, which provides for the use of a higher number of more specific categories. However, some similarities with the card sorting classification emerge as well. In the end, the new classification emerging from the card sorting exercise seems able to address some of the issues found in a previous user study [6], in which Tasker showed various usability problems. Such issues were mainly due to the high number of categories, for which it was difficult to find the elements of interest, and the use of Events and State macro-categories, which were difficult for the users to understand as well. Atooma revelaed usability issues as well because of the unbalanced distribution of the elements (most of them in the Mobile category), and the use of a graphical representation, which made it difficult to show all of them at the same time, thus making finding some of them problematic.

In terms of requirements for new solutions, this study highlights that they should facilitate the understanding of the event / condition / action paradigm, and the search and use of the elements of interest. Thus, a usable design should be able to graphically represent the cause / effect mechanism without imposing any temporal constraint regarding which to specify first. The elements of interest should be selectable from lists providing an appropriate number of elements, without having to deal with an excessive number because this makes it difficult to identify the desired elements.

References

1. Paternò, F.: End User Development: Survey of an Emerging Field for Empowering People. ISRN Software Engineering, 2013, Article ID 532659, 11 pages (2013)
2. Realinho, V., Romão, T., Dias, A.E.: An event-driven workflow framework to develop context-aware mobile applications. In: MUM 2012, vol. 22. ACM Press (2012)
3. Danado, J., Paternò, F.: Puzzle: Puzzle: A Mobile Application Development Environment using a Jigsaw Metaphor. Journal of Visual Languages and Computing **25**(4), 297–315 (2014)
4. de A. Maues, R., Barbosa, S.D.J.: Keep doing what i just did: automating smartphones by demonstration. In: Mobile HCI 2013, pp. 295–303 (2013)
5. Ur, B., McManus, E., Ho, M.P.Y., Littman, M.L.: Practical trigger-action programming in the smart home. In: CHI 2014, pp. 803–812
6. Lucci, G., Paternò, F.: Understanding end-user development of context-dependent applications in smartphones. In: Sauer, S., Bogdan, C., Forbrig, P., Bernhaupt, R., Winckler, M. (eds.) HCSE 2014. LNCS, vol. 8742, pp. 182–198. Springer, Heidelberg (2014)