Automatic Internationalization and Localization Based on Android Location Services

Aiman M. Ayyal Awwad[1] and Nur EL-Din EL- Rez[2]

[1] Computer and Information Technology Department, Tafila Technical University, Tafila, 66110, Jordan

[2] Institute of Software Technology, Graz University of Technology, Graz, 8010, Austria

Abstract

Android operates on many smartphones in many locales. To reach the most users, the app should handle all resources such as text strings, layouts, graphics, and any other static data that the app needs, in ways proper to the locales where the app will be used. Doing that requires internationalization and localization of the app to support multiple languages. In this paper, we present an approach for localizing the Android app according to the location data that the application received from the device. In particular, the proposed feature automates the locale detection process using Android's location services. Depending on the locale detected the resources' contents are rendered in a particular language used in that locale. The testing results show that the proposed feature triggers the restarts of the app with the language spoken in that locale, updates the visible user interface properly, and delivers a more personal and context-rich user experience.

Keywords: *Android, Internationalization, Localization, Location-Based Services, Mobile Applications, Smart phones.*

1. Introduction

Mobile phones and apps play an integral part in our everyday life. A tendency toward mobile application development has increased in the past few years. In order to compete in global markets, mobile app development companies need to publish world-ready products. If an app is going to be released in various countries or regions, it should support the languages of the target market. This asks for Internationalization (I18n) and Localization (L10n) of the application [1].

Software internationalization and localization are vital steps in distributing and deploying software to various countries of the world [1, 2]. Users feel more comfortable and productive if the app talks to them in their native language and shows their cultural values [3]. Every country or region has its own language, customs, and culture; accordingly, the mobile app should be localized to meet all the expectations and needs of local users and thereby increase the total number of downloads and revenue [2].

Localization of a product requires that the product is adapted to both the language and the culture of the particular market [2]. The higher an app is ranked in search results the higher the number of potential customers downloading the app. Therefore, mobile publishers should be aware of the app marketing principles in order to get the attention of customers in new markets.

As of the first quarter of 2018, Google's Play Store holds 3.8 million apps either commercial or free for public download. While the Apple's App Store holds 2 million apps for download (see Fig. 1) [4]. This huge amount of apps to choose from for the main mobile platforms and the number of total downloads imply that there must be distinguishing features to make one app more interesting and usable than others [5, 6].

According to data collected by Google and Admob in March 2014, the number of users who have stopped using an app due to a lack of localization in selected countries varies between 34% and 48% depending on the country these data were collected (United States, China, Japan, United Kingdom, and South Korea) [7]. This shows that there is definitely a need for localization which is also reflected by the general IT industry's planned investment priorities for the years 2015-2017, where localization is among the top ten [8].

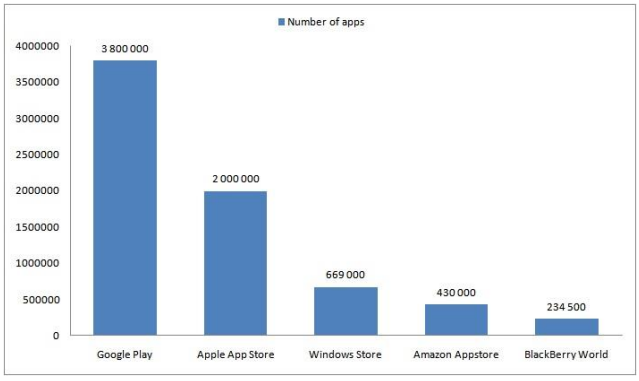

Fig. 1 Available apps in leading app stores.

In recent years, with the rapid advancements of wireless communication technology such as Global Positioning System (GPS) and Wi-Fi technology, Location-Based

Services (LBSs) have become very common. LBSs are becoming vital on smartphones. Many people using LBSs to find where they are, track their devices remotely if they are lost, find nearby amenities, and get routing directions [9]. At the same time, Internet giants, such as Google, Facebook, Twitter, and Yahoo, have a rapid expansion in LBSs, which, in turn, make the user experience becoming better and better [10].

The LBSs of Android platform provide access to tools that can be utilized to find the device's current location. This information can be used for a wide variety of purposes and can let a smartphone and the application that runs on it to have a better knowledge about its surroundings and present a richer user experience [11, 12].

Therefore, in this paper, we employ the LBSs in localizing any Android app from the original language to other languages according to the location information that the app received from the device. The app's language and locale will be switched without switching the smartphone's interface language on the system level.

2. Internationalization and Localization

The mobile App Store allows users anywhere in the world to access any desired mobile app. Therefore, software companies need to develop world-ready apps that need to comply with the local user culture and the language of a particular market, which requires internationalization and localization of the app.

In software development, internationalization and localization are activities of adapting applications for locales of other regions, and cultures [2]. Internationalization is the process of re-engineering a software product so that it can be easily adapted to different languages, regions, and cultures without further modification. While localization is the process of customizing an internationalized software for use in a specific geographical region or a specific target market. Typically, this process includes translating all original language strings to the target language and modifying the GUI so that it can be appropriate for the local market as shown in Fig. 2 [1].

Furthermore, localization is the process of providing the appropriate resources for the product based on the device's language settings and making sure the product meets the local user's expectations in terms of language, cultural identity, features, and user experience. It aimed at developing an accessible, usable, and culturally suitable product for a particular locale [2].

It is difficult enough to produce a mobile app with a good usability for the home market without worrying about potential users in other locales or cultures. Adjusting layout, graphics, colors, and menus for other regions and languages should be performed by a person who is familiar with the cultural and linguistic requirements of the particular country. Whether this person is a translator, designer, or developer does not matter. It is even better to let this process be performed by a usability engineer who knows best about software ergonomics and cultural variations.

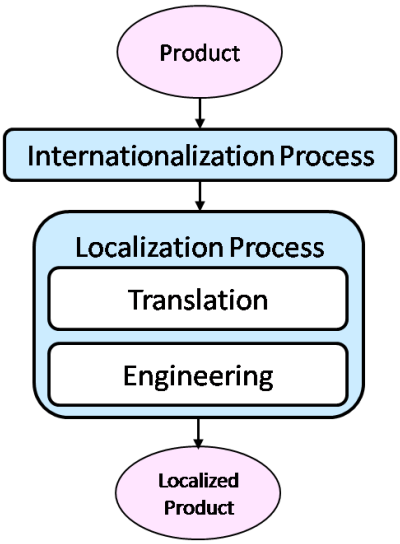

Fig. 2 Internationalization and localization processes.

In software design especially for mobile applications, it is important to consider the features and attributes of different languages. Some of the issues that the internationalization process needs to implement include [2, 13, 14]:

- Time and dates formats
- Measurements formats
- Calendar formats
- Phone numbers
- Address formats
- Currency format
- User interfaces and printed documentation
- Text directionality (left-to-right script vs. right-to-left)
- Language character encoding sets for textual display
- Names and titles
- Collation and sorting rules
- Case conversion

Similarly, the localization process may include [2, 13]

- Language translation
- Spelling variants (localization, e.g., en-US, en-CA) vs. localization, e.g., en-GB, en-AU)
- Visual language (logos and icons which have their own particular meanings)
- Locales (e.g., number formats. In the U.S. (1,234.56). In Germany (1.234,56))
- Aesthetics (layout direction)
- Cultural values and social context

The pictures are another issue to consider in the internationalization of the app if they express subtle cultural connotations within them. Pictures or images may have particular negative meanings that may offend customers. In the course of localizing software, any kind of GUI elements that might be changed according to the locale, such as text, images, media, or styles, should be declared as resources and externalized from the software's source code [2]. When the app is localized, the source code does not need to be changed for each locale, but only the app's resources corresponding to the destination locales.

As with pictures, symbols are also another issue that may cause problems in the localization of the app. For example, icons which represent hand gestures such as an OK sign or V-sign may express different meanings in different cultures [1]. However, Western symbols do not always express the same meaning abroad that they have for the local audience. Furthermore, the depiction of animals in icons may cause localization issues outside the home market.

Moreover, colors may represent cultural meanings that need to be analyzed in the app's localization. Picking the incorrect color for the app's logo or background might have destructive consequences. When localizing the mobile app into any locale, it is essential to handle a catalog of images, logos, and colors to make sure they are culturally suited [1, 15].

3. Location-Based Services

One of the prominent factors in the smartphones attractiveness is their LBSs. It is inconceivable to have a pocket device that can inform you exactly where you are in the world. LBSs are becoming increasingly significant in the world of mobile development. Such apps now make use of location information to present a richer user experience [9, 11].

Android provides some of the most appealing APIs that allow a user to determine, contextualize, and map physical locations. The location-related APIs and capabilities that the Android platform provides are utilized by developers to release great location-based applications [11].

Discovering a smartphone's location allows the developers to add an advanced functionality to a wide range of apps. For instance, applications that help users track daily exercises when hiking, biking, or running; notify them when interesting activities and services are nearby and where they are located; enable them to track their devices remotely; record current time, distance, speed, and other metrics; and more [16].

3.1 Determining a Device's Current Location

One of the main factors for the increase in popularity noted for smartphones is their portability; therefore, mobile app developers often have to retrieve the device's current location. Location information is a key part of apps such as map, camera, and social media apps to add another aspect to the information they are already handling. For mobile developers who have decided to get the current location of a smartphone and track the location of a smartphone while it moves, Android supports a fairly robust API to its location service [11].

3.2 Android Location API Components

Location-based service is a term that describes the various technologies the user can employ to discover a device's physical location. The majority of the classes that the developer will frequently use when working with location data in Android are located in the location package. Fig. 3 shows a high-level summary of how the location's components relate to each other. The components of that package are:

Location Manager

LocationManager is the entry point into the location services on Android. The LocationManager enables an app to retrieve the device's current location, track movement, inform Android when it is interested in obtaining updated location status (i.e., set up location update listeners), find available LocationProviders, and monitor the status information of the GPS receiver. The LocationManager can also provide information about the last known device's location [11, 16].

Location Provider

The LocationProvider's component is an abstraction for the different technologies that are used to determine the device's current location in Android. Though each provider produces location data differently, they all provide similar data to an app in the same way. Depending

on the device, each technology, available as a LocationProvider, offers various capabilities - including variations in power consumption, accuracy, and the ability to find elevation, speed, or bearing information [11, 12].

Location

Location's component encloses the device's current location data provided by a LocationProvider to a mobile app. It carries the quantifiable data such as latitude, longitude, and altitude. Once an app has handled a Location object, it can start the processing operation over that data [11].

Criteria

The Criteria component is used in a mobile app to query the LocationManager for LocationProviders that support specific requirements. This is helpful when an app is less interested in which LocationProvider is used and more interested that LocationProviders have some basic requirements. Practically, an app can set/unset attributes on a Criteria component to utilize the requirements of the chosen LocationProvider. In most situations, it is not recommended that the user can explicitly choose which LocationProvider to use. It is better to specify the app's requirements and allow Android to determine the best technology to use. The Criteria component is usually used to dictate the requirements of a provider in terms of accuracy, power use, and the ability to retrieve the elevation, speed, and bearing information [11, 12].

Location Listener

The LocationListener interface includes a collection of callback methods that are called in response to variations in a device's current location or variations in location's service state [11].

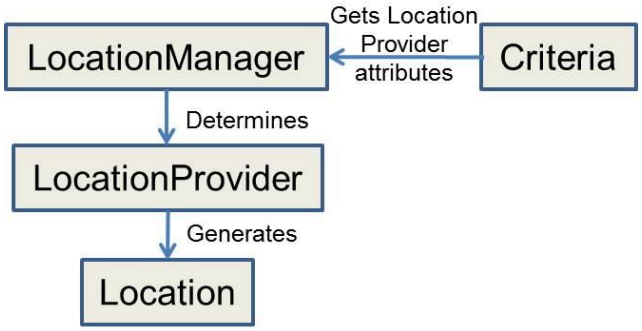

Fig. 3 Android's location package components [11].

3.3 Selecting the Appropriate Location Provider

This section provides the technologies that enable the user to find the smartphone's current location such as GPS and cell- or Wi-Fi-based location sensing techniques. The user can determine explicitly which technology to use by name, or provide a group of criteria in terms of accuracy, cost, and other requirements and permit Android to pick the most appropriate.

GPS Location Provider

Nowadays, mobile phones are usually equipped with a GPS sensor. Due to the many satellites which are orbiting the earth, you can use a GPS sensor to determine user's location easily. The GPS system operates with 27 satellites in space known as the Global Navigation Satellite System. Particularly, 24 of the satellites are active and three are backups. Each satellite revolves around the Earth every 12 hours regularly broadcasting the changing position data. The smartphone can determine a location in the world in longitude and latitude by performing computations on data from at least three of these satellites [11].

Location-based services use latitude and longitude to pinpoint physical locations. Latitude represents locations on the Earth in terms of "up or down". Latitude is the angular distance of a location on the Earth north or south of the Equator. The latitude of the Equator is 0°, the latitude of the North Pole is 90° N (north), and the latitude of the South Pole is 90° S (south). On the other hand, longitude represents locations on the Earth in terms of "left and right". Longitude is the angular distance of a point's meridian from the Prime Meridian. Lines of longitude are often referred to as meridians. The longitude of the Prime Meridian is 0° and the longitude of the Antimeridian is 180° as shown in Fig. 4 [12, 16].

In addition to latitude and longitude, which represent the vertical and horizontal location on the Earth, some geospatial sensors, including location providers in Android, can also express other metrics such as elevation, bearing, and speed. Elevation represents altitude or height above sea level on the Earth. The bearing expresses the direction (in degrees east of true north) from one's current location to another point or a person's manner of standing or moving. Speed means the rate at which something moves or how fast something is moving over the ground [11, 16].

The GPS location provider uses orbiting satellites and time to find the current location of a device and provide accurate location information. However, GPS provider needs a clear sky to operate and therefore does not always operate indoors or where satellites cannot penetrate (such as a tunnel through a mountain or underground

environments). GPS is problematic when used in an urban environment where high buildings can cause signal problems. Nevertheless, because it depends on a separate radio, the GPS provider can also consume lots of battery power more than other location providers [11].

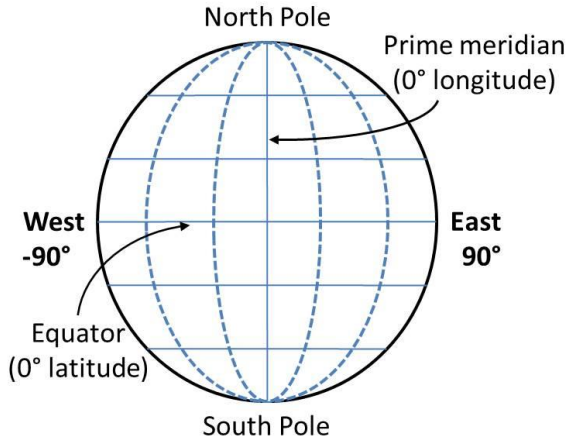

Fig. 4 Latitude, longitude, and coordinate system grids represented on a globe.

Cell-tower Location Provider

Another efficient means to discover a smartphone's current location is through cell tower triangulation. When a mobile phone is turned on, it is regularly in contact with a cell tower enclosing it. By knowing the identity of the tower that a device is currently connected to, it is possible to interpret this information into a physical location by connecting to several databases storing the cell towers' identities and their precise geographical locations. The advantage of cell tower triangulation is that it operates indoors; but, it is not as accurate as GPS. Cell tower triangulation operates best in dense areas where the cell towers are closely placed [11, 16].

Wi-Fi Network Location Provider

The third technology for discovering the location of a device is to use Wi-Fi triangulation. In practice, the device records what Wi-Fi access points can discover and the current signal strength of those access points. However, the device connects to a Wi-Fi network and examines the service provider against databases to define the location serviced by the provider. Further, as compared to the other technologies, Wi-Fi triangulation is the minimum accurate and often consumes less battery power than the GPS sensor [11, 12].

4. Employing Android Location Services in Localizing Mobile Apps

In this section, the design and implementation of employing location services in the localization process are presented in detail. The ability to create apps that support language switching based on an up-to-the-current-time location information enables them to present an incredible level of usability that was before not possible [11].

Since the LocationManager is the first door of the location service, the app needs to make a reference to the LocationManager and get the last known location. This can be achieved by using the following code snippet.

```
locationManager = (LocationManager)
        getSystemService(Context.LOCATION_SERVICE);
String provider = locationManager.getBestProvider(new Criteria(), true);
Location locations = locationManager.getLastKnownLocation(provider);
List<String> providerList = locationManager.getAllProviders();
double longitude = locations.getLongitude();
double latitude = locations.getLatitude();
```

To get the device's current location, we use the Android Geocoder. Geocoding enables the developer to convert a street address to its latitude and longitude coordinates. Android gives the ability to geocode and reverse geocode (translate from latitude and longitude coordinates into location name) without the necessity for a third-party library. The Geocoder classes are introduced as part of the Google Maps library. The Geocoder class provides access to two geocoding methods [12, 17]:

- Forward geocoding: determines the latitude and longitude of a location name.
- Reverse geocoding: determines the location name for a given latitude and longitude.

The Geocoding is used to translate between a location name and longitude/latitude map coordinates. The returned values for these calls are contextualized by means of a locale [12]. The following snippet of code is used to set the locale when creating the Geocoder.

```
Geocoder geocoder =
        new Geocoder(getApplicationContext(), Locale.getDefault());
```

Both geocoding methods return a list of address objects. Each address object contains as much detail as the Geocoder can able to resolve. This can include the latitude, longitude, phone number, and granular address information from country to a street and house number as illustrated in the below snippet of code.

```
List<Address> listAddresses = null;
try {
    listAddresses = geocoder.getFromLocation(latitude, longitude, 1);
} catch (IOException e) {
    e.printStackTrace();
}
String countryName = listAddresses.get(0).getCountryName();
String countryCode = listAddresses.get(0).getCountryCode();
```

To support multiple languages in an Android app, a new localization feature (to handle multi-language string resources) is created. Actually, we build up string resources based on the language-country codes which are supported by the application. These additional "values"-directories are created inside the resource folder ("res/") which folder name includes a hyphen and the ISO language code and sometimes the country code at the end, e.g. "values-en-rGB". The language codes are two-letter lowercase ISO language codes (e.g.,"en") based on ISO 639-1, while the country codes are two-letter uppercase ISO country codes (e.g., "GB") based on ISO 3166-1 [18].

To localize the Android app based upon the location data (longitude and latitude) that we (or our providers) received from the device, we need to map between the country code and its corresponding language code. In Android, a Map is an object that maps keys to values. A map cannot include duplicate keys; each key can map to at most one value as shown in Fig. 5 [19].

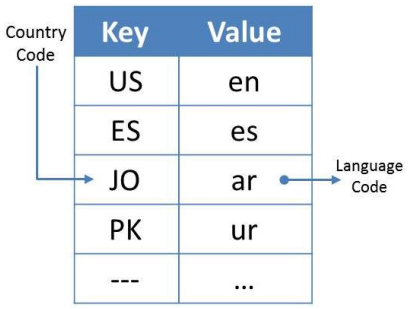

Fig. 5 The HashMap for country and language codes.

We identify a list of countries codes with their associated official language codes as shown in the snippet code below. The Map interface offers three collection views, which allow a map's contents to be viewed as a collection of keys, a set of values, or a collection of key-value mappings [19].

```
localeMap.put("US", "en");
localeMap.put("ES", "es");
localeMap.put("JO", "ar");
localeMap.put("PK", "ur");
```

Many countries, such as Canada, India, Ireland, South Africa, and Switzerland, are officially multilingual in their population. Some countries have official languages but also have regional and local official languages such as Brazil, China, Mexico, Russia, Spain, and Taiwan [20].

In practice, due to the reality of some countries have populations who speak multiple languages we cannot automatically convert the country code to its corresponding language code (i.e. there is no built-in method). On the other hand, the Operating System (OS) localization method may support multiple variants of a particular language for different countries (for example, en_GB vs en_US). Therefore, we need to use the MultiMap object; A Map that supports multiple values per key.

For localization purposes, we identify a set of countries and languages codes within the Multimap interface. For example, in Canada, English and French are the mother tongues of Canadians. So, we put one key "CA" with two different values in localeMap object as illustrated in the below snippet of code.

```
localeMap.put("CA", "en");
localeMap.put("CA", "fr");
```

Similarly, in Switzerland, there are four official languages spoken in different regions of the country: German, French, Italian, and Rumantsch (see code snippet) [21].

```
localeMap.put("CH", "de");
localeMap.put("CH", "fr");
localeMap.put("CH", "it");
localeMap.put("CH", "rm");
```

In the proposed feature, the below snippet of code returns the value to which the specified key is mapped, in other words, a language code associated with the country code is determined based on the location that has been received. Also, the proposed feature should list all languages and locales available in the received location.

```
Multimap<String, String> stringMap = Multilingual.localeMap();
String languageCode = "";
ArrayList<String> localesWithSameCountryCode = new ArrayList<>();
for (Map.Entry<String, String> entry : stringMap.entries()) {
    if (entry.getKey().equals(countryCode)) {
        localesWithSameCountryCode.add(entry.getValue());
        languageCode = entry.getValue();
        Log.i("value", languageCode);
    }
}
```

However, to support multiple languages on the app level it is necessary to instantiate a new locale for each supported language. A Locale object represents a specific geographical, political, or cultural region [1, 2]. The string value given (languages and country codes) will be used for setup (see code snippet).

```
Locale locationLocale = new Locale(languageCode, countryCode);
```

At app's runtime, the Android will load the appropriate resources according to the device's current location's language. An app can include multiple sets of resources, each customized for a different device configuration. When the app is developed, the default and alternative resources for the app should be created. When users run the app, the Android system automatically selects and loads the proper resources, based upon the device's current location as shown in Fig. 6.

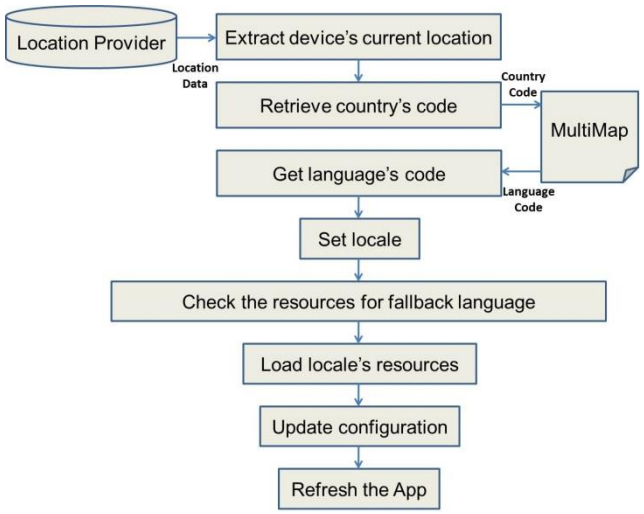

Fig. 6 The sequence steps for the proposed feature.

When the language of an app is changed, the whole app should be recreated and the currently visible UI should be updated properly according to the extracted language. So, if the app has any data object, the object's instance state must be handled (saved and restored) as shown in Fig. 7.

5. Testing Results and Discussion

The primary goal of the proposed feature is to employ the LBSs in localizing the Android App, therefore, enhancing overall user experiences. Certainly, the functionality of the proposed feature should be tested in different countries with different languages. The proposed feature is tested using the Pocket Code[1] as a reference. Pocket Code, Catrobat's version for the Android platform, is a free and open source visual programming language environment for smartphones [3].

[1] https://catrobat.org

Fig. 7 The sequence of processes for locale switching.

To evaluate the quality of the proposed approach, different mobile testers from different countries are dedicated to testing the LBS feature. We provide each of them with the APK (Android Package Kit, i.e. Android application package) file for Pocket Code. Actually, the tester's mission is to launch the product in his location and verify whether the expected and actual results are matched or not. However, if the actual locale is not the same as the expected locale, the actual locale is recorded in the "Actual locales" column. If the actual locale is the same as the expected locale, "Pass" is recorded in the "Actual locales" column to indicate that the expected and actual locales are identical as illustrated in Table 1.

Table 1: The expected and actual results for the proposed feature

Country	*Expected locales*	*Actual locales*
United States	English	Pass
France	French	Pass
Jordan	Arabic	Pass
Spain	Spanish	Multi-Locale List
Austria	German	Pass
Italy	Italian	Pass
Switzerland	German, French, Italian, or Rumantsch	Multi-Locale List

Fig. 8 (a) – (e) shows the localized versions for different languages: English, French, German, Italian, and Arabic in the selected countries: United States, France, Austria, Italy, and Jordan, respectively. During the testing phase, it is found that the proposed feature localizes the Pocket Code properly, and the product is cosmetically appropriate, linguistically correct, and culturally congruent to local customs and that no issues have been presented during the localization process.

For the countries that are officially multilingual (e.g., Switzerland), the user can tap the location icon that appears in the ActionBar and then selects her preferred locale from the list of available locales (see Fig. 8 (f)).

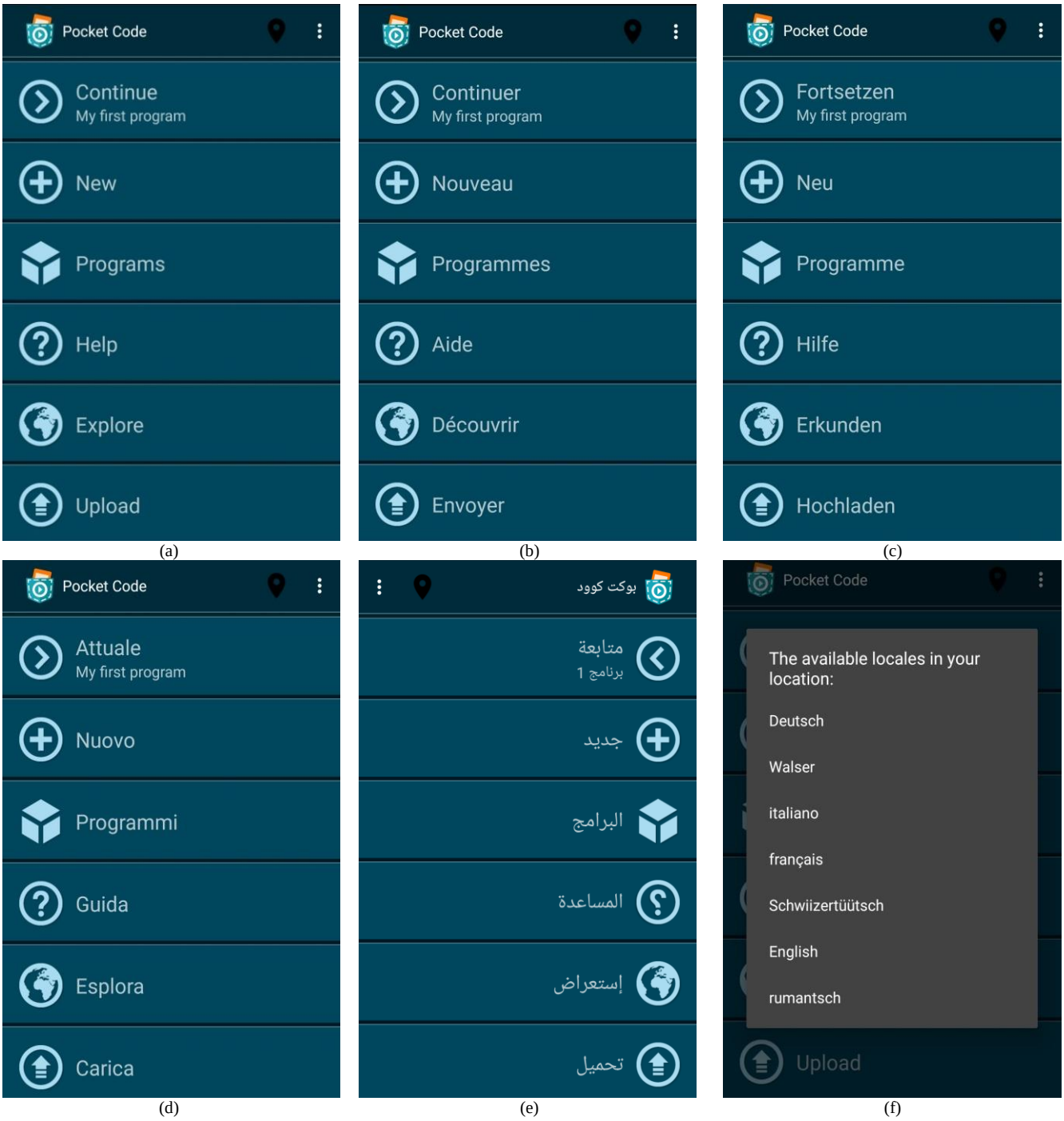

Fig. 8 Screenshots for Pocket Code's localized version (a) Origin version (b) French (c) German (d) Italian (e) Arabic (f) Available locales in Switzerland.

6. Conclusion

In this paper, a feature for localizing the Android app according to the received location data is introduced. In most of the applications, it will be more useful to present to users an option for location-based language switching. It will not affect the default language settings for the device and will not make any issue for other applications on the user's device. Using location sensor, locale switching feature is enriched with location information which allows the Android app to be correctly localized and present an incredible level of usability. The testing results show that the proposed feature localizes the Android app properly without any issues that may be produced during the localization process and after automating internationalization and localization process there will not be a language barrier for the mobile apps users.

Acknowledgments

The authors would like to thank Professor Wolfgang Slany and the Catrobat development team[2].

References

[1] Aiman M. Ayyal Awwad, "Localization to Bidirectional Languages for a Visual Programming Environment on Smartphones", International Journal of Computer Science Issues (IJCSI), Vol. 14, No. 3, 2017, pp.1.

[2] Aiman M. Ayyal Awwad, and Wolfgang Slany, "Automated Bidirectional Languages Localization Testing for Android Apps with Rich GUI", Mobile Information Systems, Vol. 2016.

[3] Luhana, Kirshan Kumar, Christian Schindler, and Wolfgang Slany. "Streamlining mobile app deployment with Jenkins and Fastlane in the case of Catrobat's pocket code", IEEE International Conference on Innovative Research and Development (ICIRD), Bangkok, 2018, pp. 1-6.

[4] Number of apps available in leading app stores as of 1st quarter 2018, The Statistics Portal, Statista. Accessed: 2018-07-05. [Online]. Available: https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/

[5] Cumulative number of apps downloaded from the Google Play as of May 2016 (in billions), The Statistics Portal, Statista. Accessed: 2018-04-05. [Online].Available:https://www.statista.com/statistics/281106/number-of-android-app-downloads-from-google-play/

[6] Cumulative number of apps downloaded from the Apple App Store from July 2008 to June 2017 (in billions), The Statistics Portal, Statista. Accessed: 2018-03-12. [Online]. Available: https://www.statista.com/statistics/263794/number-of-downloads-from-the-apple-app-store/

[7] Share of app users who have stopped using an app because it was not localized properly as of March 2014, The Statistics Portal, Statista. Accessed: 2018-04-15. [Online]. Available: https://www.statista.com/statistics/296304/mobile-app-abandoment-rate-due-to-lacking-localization/

[8] Top information technology (IT) priorities over the next two years worldwide as of late 2015, The Statistics Portal, Statista. Accessed: 2018-04-15. [Online].Available: https://www.statista.com/statistics/529314/worldwide-survey-it-top-priorities/

[9] C. Ferreira, L. F. Maia, C. Salles, F. Trinta, and W. Viana, "A Model-based Approach for Designing Location-based Games", 16th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames), Curitiba, 2017, pp. 29-38.

[10] Mingjie Ma, "Enhancing Privacy Using Location Semantics in Location Based Services", IEEE 3rd International Conference on Big Data Analysis (ICBDA), Shanghai, 2018, pp. 368-373.

[11] Greg Milette, and Adam Stroud, Professional Android™ Sensor Programming, England: Wrox, 2012.

[12] Reto Meier, Professional Android 4 Application Development, England: Wrox, 2012.

[13] N. Kotze, "Internationalization and Localization Testing", Testing Experience: Magazine for Professional Testers, No. 27, 2014, pp. 16-19.

[14] C. Kopsch, "Localization testing one-year status report for a localization project", Testing Experience: Magazine for Professional Testers, No. 27, 2014, pp. 20-22.

[15] Aiman M. A. Awwad, C. Schindler, K. K. Luhana, Z. Ali, and B. Spieler, "Improving Pocket Paint Usability via Material Design Compliance and Internationalization & Localization Support on Application Level", Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services. ACM, Vienna, 2017, Vol. 99, pp.1-8.

[16] Charlie Collins, Michael Galpin, and Matthias Kaeppler, Android in Practice, United States: Manning Publications, 2011.

[17] Wei-Meng Lee, Beginning Android™ 4 Application Development, England: Wrox, 2012.

[18] Locale, Android Developers, Accessed: 2018-04-10. [Online]. Available: https://developer.android.com/reference/java/util/Locale

[19] Map, Android Developers, Accessed: 2018-05-15. [Online]. Available: https://developer.android.com/reference/java/util/Map

[20] List of multilingual countries and regions, Accessed: 2018-06-12. [Online]. Available: https://www.revolvy.com/page/List-of-multilingual-countries-and-regions

[21] Switzerland's Four National Languages, Markus G. Jud, Accessed: 2018-06-12. [Online]. Available:

[2] http://developer.catrobat.org/credits

http://official-swiss-national-languages.all-about-switzerland.info/

Aiman Mamdouh Ayyal Awwad is currently a full-time lecturer in the Department of Computer Science and IT at Tafila Technical University. He received his B.Sc in Computer Science from Mutah University in 2007 and his M.Sc in Computer Science from the University of Jordan in 2010. He obtained his Ph.D. in Computer Science from Graz University of Technology/ Austria in 2017 with research interests related to smartphone applications. From February 2010 to September 2014, he was a lecturer at Computer Science and IT Department / Tafila Technical University. He has more than 7 publications in various international journals and conferences. His research interests include mobile computing and applications, image processing, and cellular automata.