Adaptative Applications for Heterogeneous Intelligent Environments

Aitor Almeida[1], Pablo Orduña[1], Eduardo Castillejo[1], Diego López-de-Ipiña[1], and Marcos Sacristan[2]

[1] DeustoTech –Deusto Institute of Technology, Universidad de Deusto
Avenida de las Universidades 24,
48007 Bilbao Spain
{aitor.almeida,pablo.orduna,eduardo.castillejo,
dipina}@deusto.es
[2] Parque Tecnológico de Asturias Parcela 30
E33428 Llanera Asturias Spain
marcos.sacristan@treelogic.com

Abstract. As the device ecosystem in the intelligent environments becomes more complex, the need for supporting a wider range of devices becomes crucial. To address this challenge, this paper describes a framework, Imhotep, which can be used to develop mobile applications, easily adapting them to the constraints imposed by the user and device capabilities through a set of preprocessor directives. These directives are enhanced with concepts that are automatically adjusted to the current trends of mobile devices by using a fuzzy knowledge-eliciting reasoner.

Keywords: Adaptation, intelligent environments, fuzzy logic.

1 Introduction

Modern homes host a technological ecosystem composed by heterogeneous devices with a wide range of capabilities and characteristics. Developing applications for these kind of scenarios presents a set of challenges that make difficult for developers to create applications that can reside in any of these devices. This problem reaches new dimensions when developers start considering not only the different devices, but also users. Users also possess a wide variety of abilities (sensorial, cognitive…) that make difficult to target every group with one application. In our research group's[1] previous work our main focus has been the development of server-side applications [1, 2] to create intelligent environments that react to the changes in the context. In these projects we have neglected the client side of the intelligent environment applications, but we acknowledge that users are a central element in intelligent and assistive environments. The devices employed by users to interact with the environment are as diverse as the users themselves, providing a wide range of possibilities.

[1] http://www.morelab.deusto.es/

B. Abdulrazak et al. (Eds.): ICOST 2011, LNCS 6719, pp. 1–8, 2011.

To tackle this problem we have created a framework for developers that facilitates the creation of applications adapting them to the device that are going to run into. The framework allows developers to state how their applications should react to the different device characteristic. What is more, it simplifies the development process, so developers without previous experience with mobile devices can use more natural expressions to define the capabilities of the devices. To do so the framework uses a fuzzy inference engine along trend metrics to characterize the mobile devices. The framework also provides an application server that stores the program and adapts it to the target device when a new installation request is received. We have previously employed this approach [12] to create user interfaces adapted to the capabilities of the user. This solution is by some means similar to the different repositories developed by the mobile operating system manufacturers. Apple started the App Store in July 2008, Android Market was available to users in October 2008, Blackberry published App World in April 2009, Nokia launched Ovi Store in May 2009 and Microsoft released the Windows Phone Marketplace in October 2010. Third-party developers can use these software repositories to upload their applications and publish them to the final customers. Applications have to deal with its own customization dynamically, checking the constraints of the devices on which they are being executed. Therefore, developers cannot upload a customized build for each user optimizing the bandwidth and the space on the phone, in contrast to the approach presented in this paper.

2 Related Work

Application adaptation has already been used in other areas. In [4, 5], authors developed a middleware to adapt services to context changes based on a user-centric approach. In [6], authors describe a mechanism to adapt the user interface to some physical characteristics of the user like his height or the distance to the screen, without a special emphasis to the disabilities. In [7], the authors defined a context model for information adaptation that takes into account the user needs and preferences.
In order to select one approach for the creation of adaptative applications we have analyzed the techniques used by different frameworks: Custom mark-up languages (as those used by OpenLaszlo [8]), use of factories (like Google Web Toolkit [9] and EMI2lets [3]) and the Preprocessor Directives (as used in Antenna [10] and J2ME Polish [11]). All these alternatives have their own strengths and weaknesses. After analyzing the alternatives we have concluded that the approach used by Antenna and J2ME Polish is the most fitting one for our requirements. Decoupling the framework from the programming language provides a versatility not attainable with the other approaches, making the framework suitable to be used in the development of any application. What is more, because there is no need to learn a new language any developer can easily include the necessary directives in his code.

3 Adaptative Application Framework

This section describes the architecture. The Imhotep framework is divided into two general elements: the application server and the mobile client (see Fig. 1). The mobile

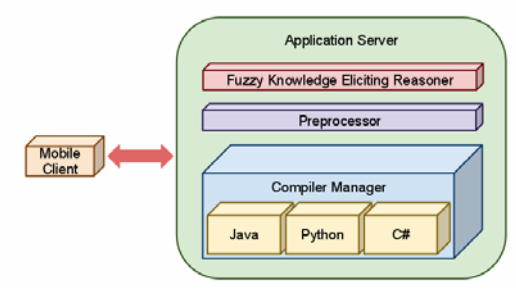

Fig. 1. The adaptative framework is divided in two main elements, the mobile client and the application server

client will perform queries on the application server to search applications available in the repository. With a selected application, the mobile client will request it, providing its characteristics, and the application server will return a customized binary file that the mobile client will deploy. In order to build this customized binary file, the application server will first use the Fuzzy Knowledge Eliciting Reasoner to infer new values taking into account the rules that the developer provided. For instance, the developer could establish concepts such as *"if the memory is higher than 256 MB and the frequency of the CPU is higher than 512 MHz, then COMPUTING_CAPABILITIES are HIGH"*. With these inferred values, as well as with the non-inferred values that the system already had of the device, the application server will use the Preprocessor to parse the application source code. The non-inferred values are directly taken from the WURFL[2] (Wireless Universal Resource File) database, version 2.9.5. WURFL is an open source database containing data about features and capabilities of a large number of mobile devices. We have extracted those relevant variables from the database and imported in the Imhotep one. Finally, the application server will use the Compiler Manager to build the adapted application with the generated source code. Given that these compilations are cached, the full process is executed once per each application and different configuration so when the application had already been compiled the time used is not significant.

3.1 Preprocessor Directives

The code provided by the developer will be annotated with a set of directives that the preprocessor will compute. These directives define how the final source code must be generated, providing conditions for certain regions of code to be added or skipped, and adding Imhotep variables that the preprocessor will adapt for each compilation. Therefore the preprocessor will parse the original annotated source code and will generate the code that will actually be compiled. The preprocessor identifies the directives when they start by //# in languages that support inline comments starting by //, such as Java, C# or C++, #// in languages that support inline comments starting by #, such as Python or Perl, and '// in VB.NET.

[2] http://wurfl.sourceforge.net/

```
//#if defined(${piramide.devices.height})
   //#if ${piramide.devices.height} > 150 * 2
          addTenRows();
   //#elif ${piramide.devices.width} > 200
          addTenCols();
   //#else
          addFiveRowsAndCols();
   //#endif
//#else
     addFiveRowsAndCols();
//#endif
```

Fig. 2. Preprocessor Directives

The preprocessor can avoid the compilation of fragments of code if certain conditions are matched. These conditions can include calls to functions provided by the system. Basic string and math functions are available, including *lowercase*, *trim*, *contains*, *round* or *sqrt*, as well as functions to check if a certain variable is available. The conditions can be embedded, as shown in Fig. 2. Whenever it is required, developers can directly store in programming variables the system ones. This way, developers can adjust programmatically to the exact values of the variables.

3.2 Fuzzy Knowledge Eliciting Reasoner

The application developer may not be aware of the current state of the device market. For instance, developers might only want to establish high level asserts such as *"is the screen big?"* or *"is the processor fast?"*. However, these asserts depend on the global market of mobile devices at a particular moment. What was considered a big resolution for a mobile device in 2005 is considered a medium screen nowadays, and a fast processor in 2005 might become average or slow today. Furthermore, developers might want to construct higher level asserts combining different capabilities. For example, *"video capabilities"* can be seen as the combination of the screen size, the resolution and the supported video codecs. For this reason we have developed the Fuzzy Knowledge-Eliciting Reasoner. The goal of the Fuzzy Knowledge-Eliciting Reasoner is to identify new capabilities using the already existing ones and to fuzzify them. To do this we have defined a set of fuzzy rules that take numeric values from the existing capabilities as input and create symbolic values for the new ones. An example for the reasoning that takes place in this stage would be: "*If the resolution is big and the screen size is big the video suitability is very high*". The main problem we have encountered using fuzzy rules is that we need to fuzzify the crisp variables encountered in the databases (in our case WURFL 2.9.5). This raises some challenging questions. What do we consider a *"big"* screen size? How can we identify what characteristics are inherent of the average mobile device? These concepts are relative to the values of other device models. One screen is big if its height and width are larger than the average values of the other models. But all device models can not have the same weight in the calculation, neither all the device models have sold the same number of units. This is why the most popular models should have more weight during this calculation. In order to calculate the popularity of one device we have to adjust it with its *"age"*. Popularity fades away with the passing of time.

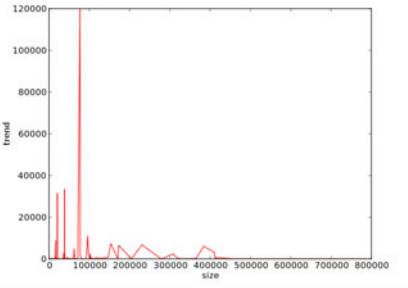

Fig. 3. Popularity calculation for all the resolutions of the mobile devices in WURFL

Users tend to change their mobile phones frequently, drastically altering the perception of what is a big screen from one year to another. Our proposed solution uses Google Trends[3] to identify the popularity of each mobile device. We use the trend information of the last three years (2007, 2008, 2009 and the first 2 months of 2010). While this number does not represent the sale volume, we think that it is a good indicator of the interest shown by the consumers in a specific model. Due to the lack of data regarding the real sale volume for most mobile devices is one of the few available indicators. This trend value can change drastically from one location to another; the most popular devices are not the same in Japan and Spain. To tackle this problem we support the geolocation of the results to filter them according to the needs of the developers.

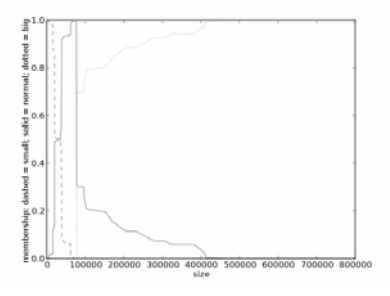

Fig. 4. Membership function for the resolution of the screen using 3 linguistic terms: small (dashed), normal (solid) and big (dotted)

Once the popularity values for all the devices in the WURFL database have been retrieved, the membership functions of the different crisp capabilities are calculated. For example, in Fig. 3 can be seen the adjusted values of popularity expressed in number of searches performed (vertical axis) for the values of the screen resolution represented as the result of multiplying the height and width (horizontal axis). As can be seen in the graph the majority of the population is gathered between 50000 and

[3] http://www.google.es/trends

100000. We use this popularity value to calculate the membership function of each variable (see Fig. 4). In this case the screen resolution has 3 linguistic terms: small (dashed), normal (solid) and big (dotted). Finally we use these membership functions in the fuzzy rules described previously in this section.

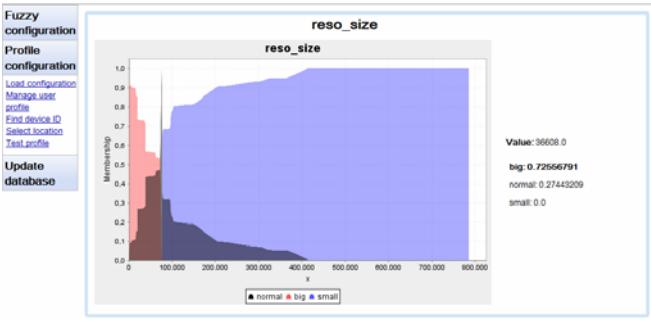

Fig. 5. Testing the generated fuzzy configuration

To ease the creation of the fuzzy rules we have developed a wizard[4] that guides developers during the process. This wizard is web based and has been developed with Google Web Toolkit [9]. Using it, developers can load an XML file with the description of the fuzzy terms, as well as the profile configuration, indicating the capabilities of the user and the mobile device used. The device is chosen from the list of supported mobile devices, taken from the WURFL database. Once the variables and rules have been created the results can be tested, providing a graphic of the generated membership functions (see Fig. 5)

4 Evaluation

In order to evaluate the usability of Imhotep, a survey was performed among software developers not involved in the project with the questions detailed in Table 1, highlighting those with highest votes for each question. At the moment of writing this paper 19 developers have participated in the survey. As it can be appreciated, 73.68% of the developers considered that Imhotep facilitates much (4) or very much (5) the development of accessible applications, and 84.21% considered that it was useful (4) or very useful (5). The time required for learning to use was considered very low (1), and developers considered the directives simple enough (1), although the time required to perform the test was considered medium (3). Finally, most developers considered Imhotep interesting (4-5) for future accessible or device adaptable applications developments.

As part of the evaluation, we have implemented a use case. Previously Imhotep had been used to adapt the graphical user interfaces [12] to the capabilities of the user, providing different interfaces for blind people and for people without visual disabilities. However, as detailed in this paper, Imhotep is also used for facilitating the

[4] www.morelab.deusto.es/imhotep/

Table 1. Evaluation results

Question	1 (%)	2 (%)	3 (%)	4 (%)	5 (%)
Previous experience in the development of accessible/device adaptable apps (1 – no experience; 5 – experience)	**63**	26	11	0	0
Time used to learn Imhotep (1 – short; 5 – long)	**37**	**37**	16	5	5
Preprocessor directives complexity (1 – low; 5 – high)	**63**	26	5	0	5
Facilitation of the development (1 – low; 5 – high)	5	0	21	**58**	16
Usefulness of Imhotep (1 – useless; 5 – useful)	0	0	16	**68**	16
Time to perform the test (1 – short; 5 – long)	26	21	**37**	16	0
Would you use Imhotep in the future? (1 – unlikely; 5 – likely)	0	5	21	**58**	16

multidevice application development for smart environments. As Fig. 6 shows, an application developed with Imhotep controls different sensors -temperature- and actuators -lights, heating- of a smart room.

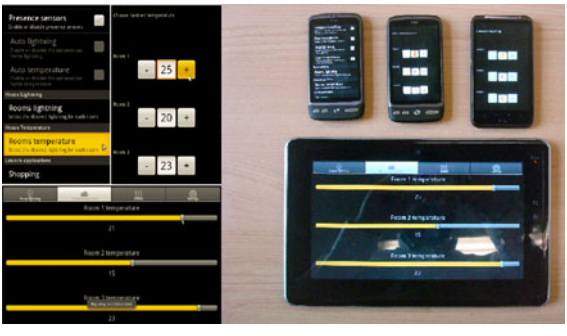

Fig. 6. The same application adapted to HTC Desire and Toshiba Folio 100

This application has been adapted to two different types of devices -a mobile phone and a tablet- so while there is a single project to maintain and the source code is the same, the downloaded binaries are different for each device. The images shown in the tablet will have a higher resolution, requiring more space in disk compared to those downloaded for the mobile phone. Since the Application Server counts with information of the memory and the microprocessor of each device, the developer can ask the application to store more or less data in the ontology used in the smart environment, so an HTC Desire would automatically store less data than an HTC Desire HD. The result is that each device will download an adapted binary of the application with the same behavior but different user interfaces, memory buffer sizes and communications technologies best adapted to the capabilities of each mobile device.

5 Conclusions

This paper has presented a framework that enables the creation of applications suited for different user and device capabilities easily. We have also discussed a fuzzy-logic based inference mechanism that allows identifying new capabilities based on those ones already known. Future work will focus on making Imhotep more developer friendly with the integration of the framework with an IDE. We will also like to merge the data from different databases. Finally we will improve all the membership function calculation process.

Ackwoledgements

This work has been supported by project grant TSI-020301-2008-2 (PIRAmIDE), funded by the Spanish Ministerio de Industria, Turismo y Comercio.

References

1. López-de-Ipiña, D., Laiseca, X., Barbier, A., Aguilera, U., Almeida, A., Orduña, P., Vazquez, J.I.: Infrastructural Support for Ambient Assisted Living. In: 3rd Symposium of Ubiquitous Computing and Ambient Intelligence, Advances in Soft Computing, vol. 51, pp. 66–75 (2009)
2. Almeida, A., López-de-Ipiña, D., Aguilera, U., Larizgoitia, I., Laiseca, X., Orduña, P., Barbier, A.: An Approach to Dynamic Knowledge Extension and Semantic Reasoning in Highly-Mutable Environments. In: 3rd Symposium of Ubiquitous Computing and Ambient Intelligence, Advances in Soft Computing, vol. 51, pp. 265–273 (2009)
3. López-de-Ipiña, D., Vázquez, J.I., Garcia, D., Fernández, J., García, I., Sáinz, D., Almeida, A.: A Middleware for the Deployment of Ambient Intelligent Spaces. In: Cai, Y., Abascal, J. (eds.) Ambient Intelligence in Everyday Life. LNCS (LNAI), vol. 3864, pp. 239–255. Springer, Heidelberg (2006)
4. Ahn, H.-I., Lee, J.-E., Yoon, Y.-I.: A middleware for user centric adaptation based on fuzzy in ubiquitous environment. In: Sixth International Conference on Advanced Language Processing and Web Information Technology (2007)
5. Abe, M., Morinishi, Y., Maeda, A., Aoki, M., Inagaki, H.: Coool: a life log collector integrated with a remote-controller for enabling user centric services. Digest of Technical Papers International Conference on Consumer Electronics (2009)
6. Hagen, S., Sandnes, F.: Toward accessible self-service kiosks through intelligent user interfaces. Pers. Ubiquitous Comput. 14(1), 1–7 (2010)
7. Hervás, R., Bravo, J., Fontecha, J.: A context model based on ontological languages: a proposal for information visualization. J. Univers. Comput. Sci. (JUCS) 16(12), 1539–1555 (2010)
8. OpenLaszlo (2010), `http://www.openlaszlo.org`
9. Google Web Toolkit. code.google.com/webtoolkit (2010)
10. Antenna. antenna.sourceforge.net (2010)
11. J2ME Polish (2010), `http://www.j2mepolish.org`
12. Almeida, A., Orduña, P., Castillejo, E., López-de-Ipiña, D., Sacristán, M.: Imhotep: an approach to user and device conscious mobile applications. Personal and Ubiquitous Computing (2011) ISSN: 1617–4909