Configuration Folding: An Energy Efficient Technique for Context Recognition

Muhammad Umer Iqbal, Marcus Handte, Stephan Wagner, Wolfgang Apolinarski, Pedro José Marrón
Networked Embedded Systems
Universitaet Duisburg-Essen
Duisburg, Germany
{*umer.iqbal|marcus.handte|stephan.j.wagner|wolfgang.apolinarski|pjmarron*}*@uni-due.de*

***Abstract*—This demonstration presents configuration folding, a novel technique which enables the energy efficient execution of multiple context recognition applications on a personal mobile device. Configuration folding achieves energy efficiency by identifying and removing the redundant functionalities in different context recognition applications at runtime. The demonstration shows the effectiveness of configuration folding. To do this, we have implemented two non-trivial sound-based context recognition applications that are able to detect speech and music. During the demonstration, we will execute these applications with folded and unfolded configurations and we will show how the resulting energy gains can be measured. Furthermore, we present how similar applications can be built by means of composition.**

***Keywords*-Context recognition, energy efficiency, component system.**

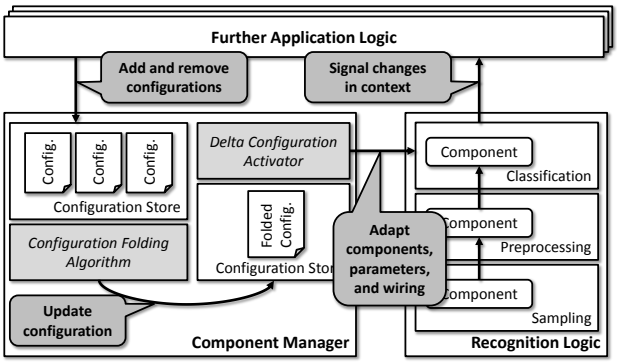


Figure 1. Component System

I. Introduction

In recent years, personal mobile devices like smart phones have evolved as a major context recognition platform. A number of applications have been developed by the research community targeting different domains including physical activities [1], social networking [2], health care [3], etc. The growing number of such applications implies that users are often involved in executing different sets of them simultaneously resulting in a considerable energy usage.

A closer analysis of context recognition applications shows that they often use similar signal processing and data mining techniques. For example, researchers have built a number of applications that detect physical activities using accelerometers. If two such applications recognize, for example, falls and stair climbing, it is highly likely that similar - if not identical - sampling and preprocessing functions are implemented as part of each of them. If both applications are executed simultaneously on the same device, this results in duplicate computation which, in turn, leads to an unnecessarily high increase in energy usage.

The increased energy utilization stemming from duplicate application functionality can be reduced by manually integrating the different context recognition applications. However, on the one hand, the required overhead for such a manual integration increases fast with the number of target applications. On the other hand, manual integration requires internal knowledge about the applications and thus, it prevents isolated application development. Consequently, this approach is not only tedious but also inherently limited in scale.

The basic idea behind configuration folding is to avoid this problem by introducing a component structure that facilitates runtime analysis of context recognition applications. By exploiting the component structure, configuration folding can automatically detect and remove redundant application functionality at runtime. To do this, configuration folding rewires the component configuration of each executed application in such a way that samples and (intermediate) results are shared among them.

II. Approach

In the following, we briefly outline the overall approach taken by configuration folding. A more detailed technical description of the approach and an experimental evaluation can be found in [4]. In addition to that, more details on the underlying system can be found at [5].

A. Component System

In order to enable the runtime analysis of arbitrary context recognition applications, we have developed a minimal component system for personal mobile devices. As depicted in Figure 1, the component system introduces a separation between the context recognition logic and further application logic such as user interfaces or networking in order to avoid restrictions on the latter. The main building blocks of the recognition logic realized by the component system

are components, connectors and configurations. Components represent atomic and reusable functions implemented at developer defined granularity. However, usually each application functionality is implemented as a single component, e.g. filtering, FFT, windowing, etc. Connectors are used to model the interaction between components in terms of data and control flow. Thus, they implicitly define the execution order. Configurations represent a description of the context recognition logic of a particular application. Consequently, they consist of a description of the components required by the application as well as the associated connections between them.

B. Configuration folding

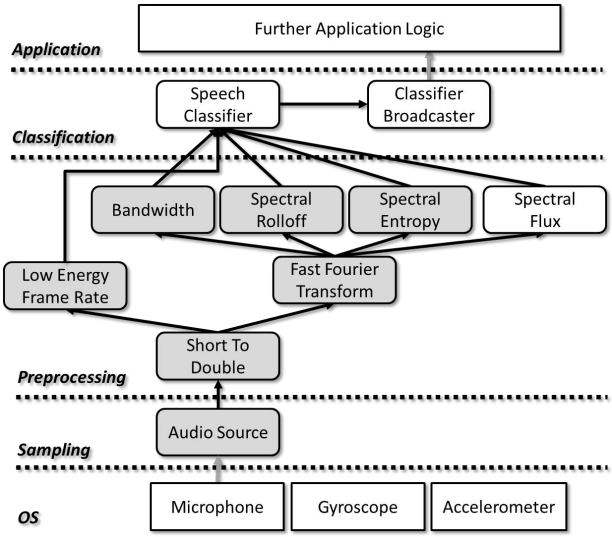
Configuration folding analyses the application's structure by means of its configuration. It identifies redundant components that are present in different applications and derives a folded configuration. The folded configuration contains components and connectors that jointly realize the functionality of the input applications without having redundancies. As depicted in Figure 1, this single folded configuration is then used by the component system to execute all of them. Finding redundancy between components does not merely consist of finding out the exact components in different configurations but it also involves ensuring that set of input components are also identical.

III. Demonstration

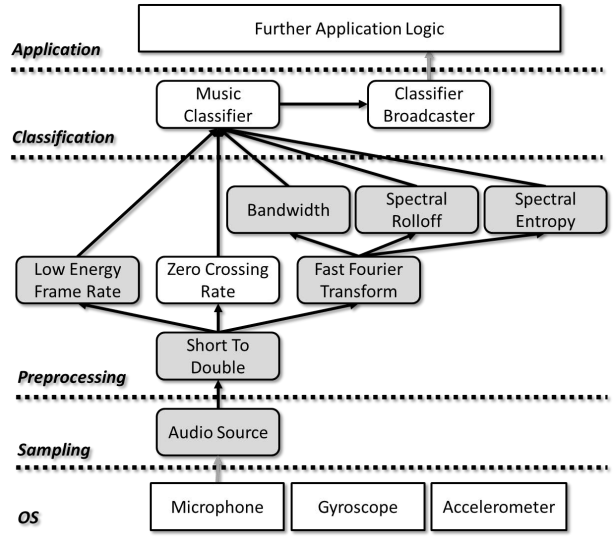
In summary, the demonstration consists of three main building blocks. First, we demonstrate two sound-based context recognition applications that are executed on mobile phones. Second, we demonstrate how such applications can be built using the component system. Third, we demonstrate configuration folding and we show how its effectiveness can be determined.

A. Example Applications

To analyse the effectiveness of configuration folding, we have implemented two non-trivial context recognition applications for music and speech detection based on the descriptions given in [6]. During the demonstration, we present both applications. The recognition logic implemented by each of them is depicted Figure 2(a) and Figure 2(b), respectively. Both applications require a number of components. At the lowest layer, they require a sampling component to capture data from a microphone. On top of that, there are various preprocessors that compute features in both, the time as well as the frequency domain. Examples for the former are the low energy frame rate or the zero crossing rate. Examples for the latter are the spectral entropy or the spectral rolloff. Finally, on top of that there are specialized classifier components that determine the sound class based on the underlying features. As indicated by the grey color, both applications share a number of redundant



(a) Componentized Speech Recognition Logic



(b) Componentized Music Recognition Logic

Figure 2. Application Example

components. This can be exploited to reduce the energy usage. However, as they are performing different tasks, they also exhibit components that are specific to each application. Consequently, none of the applications would be able to replace the other one.

B. Component System

To show how similar applications can be built using the component system, we demonstrate application development using a set of Eclipse-based tools. The tools enable the composition of new applications by wiring and parametrizing different components. By adapting the parametrization of different components and by modifying the wiring of an ap-

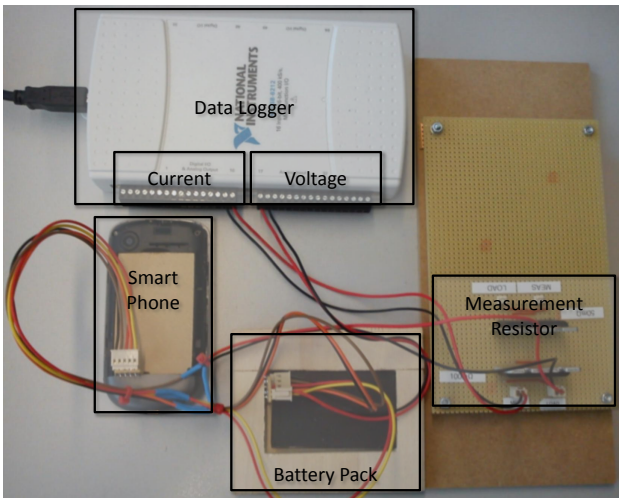
plication, it is possible to fine-tune or modify the application behavior without implementing additional program logic. Using the source code of the example applications described previously, we demonstrate the benefits and limitations of application development using the system.

C. Configuration Folding

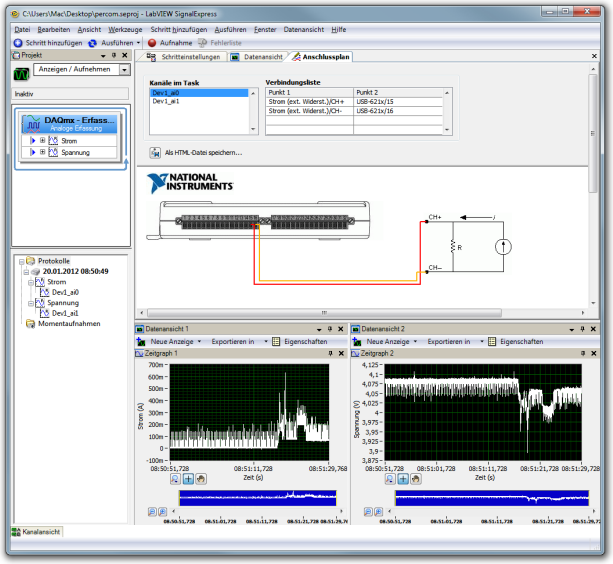
To demonstrate configuration folding, we apply it on the two applications described previously. Furthermore, we show how to measure the improvements in energy usage. To do this, we use a number of mobile phones that are running Android. On one phone, we execute speech detection application and on the second, we execute music detection application. On another phone, we execute the resulting folded configuration which combines the other two applications in an energy-efficient manner.

In order to measure and visualize the actual energy consumptions, we have developed a energy measurement setup based on the descriptions given in [7]. As shown in Figure 3(a), we connect the mobile phone battery to a high speed data logger. The resulting setup measures the instantaneous current and voltage readings when the applications are running as depicted in Figure 3(b).

As described in [4], a detailed analysis of the presented applications shows that configuration folding can save up to 48 percent of energy. The reason for this is twofold. First, due to the configuration folding, there are no redundant computations. Second, due to the joint sampling for both applications, the sensors exhibit lower duty cycles in typical application scenarios.



(a) Power Measurement Setup



(b) Power Measurement Application

Figure 3. Power Measurements

ACKNOWLEDGEMENTS

This work has been partially supported by LIVING++ funded by the BMWi under contract number KF2095019FR0 as well as CONET and GAMBAS, both funded by the European Commission under FP7 with contract numbers FP7-2007-2-224053 and FP7-2011-7-287661, respectively.

REFERENCES

[1] J. Lester, T. Choudhury, and G. Borriello, "A practical approach to recognizing physical activities," in *4th International Conference on Pervasive Computing*, 2006, pp. 1–16.

[2] E. Miluzzo, N. D. Lane, S. B. Eisenman, and A. T. Campbell, "Cenceme: injecting sensing presence into social networking applications," in *Proceedings of EuroSSC'07*, ser. EuroSSC'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 1–28.

[3] J. E. Bardram, "Applications of context-aware computing in hospital work: examples and design principles," in *2004 ACM symposium on Applied Computing*, ser. SAC '04, 2004.

[4] U. Iqbal, M. Handte, S. Wagner, W. Apolinarski, and P. J. Marron, "Enabling energy-efficient context recognition with configuration folding," in *IEEE International Conference on Pervasive Computing and Communications*, Lugano, Switzerland, 2012.

[5] Networked Embedded Systems Group, University of Duisburg-Essen, "Narf adaptive recognition recognition framework homepage," 2012. [Online]. Available: http://www.narf.mobi

[6] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell, "Soundsense: scalable sound sensing for people-centric applications on mobile phones," in *Proceedings of MobiSys '09*, ser. MobiSys '09. New York, NY, USA: ACM, 2009, pp. 165–178.

[7] A. Rice and S. Hay, "Decomposing power measurements for mobile devices," in *IEEE International Conference on Pervasive Computing and Communications*, 29 2010-april 2 2010, pp. 70 –78.