This is a pre print version of the following article:

A Self-Reconfigurable Framework for Context Awareness / Bicocchi, Nicola; Fontana, Damiano; Zambonelli, Franco. - ELETTRONICO. - (2014), pp. 1-7. (Intervento presentato al convegno IEEE Symposium on Computers and Communication tenutosi a Funchal, Madeira, Portugal nel 23-26 June) [10.1109/ISCC.2014.6912485].

IEEE
Terms of use:

The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

24/04/2024 02:04

(Article begins on next page)

A Self-Reconfigurable Framework for Context Awareness

Nicola Bicocchi[1], Damiano Fontana[2], and Franco Zambonelli

[1] Dip. di Ingegneria Enzo Ferrari
Universitá di Modena e Reggio Emilia
Modena, Italia
[2] Dip. di Scienze e Metodi dell'Ingegneria
Universitá di Modena e Reggio Emilia
Reggio Emilia, Italia
name.surname@unimore.it

Abstract. Urban environments are increasingly pervaded by ICT devices. Soon, citizens and technologies could collaboratively constitute large-scale socio-technical organisms supporting both individual and collective awareness. This paper illustrates a modern awareness framework designed to deal with the complexity of this scenario. The framework is able to collect and classify data streams in a modular way by supporting service oriented, reconfigurable components. Furthermore, we evaluate an innovative meta-classifcation scheme based on state-automata for *(i)* improving energy efficiency, *(ii)* improving classification accuracy and *(iii)* improving software engineering of aware systems, without affecting the overall performance.

Key words: Awareness Framework, Meta-Classication, Self-Reconfiguration, Self-Optimization

1 Introduction

The widespread adoption of sensor networks, actuators and computational resources capable of interacting with people is transforming urban environments [2, 17]. Citizens will be continuously connected in both a situation-aware and socially-aware way [19]. This will eventually contribute to define a dense ecosystem whose individual components will enable collaboration between ICT devices and humans, enabling advanced urban services ranging from transportation systems, to environmental sustainability and participatory governance [10, 1].

In this scenario, awareness modules are called for knowledge collection, organisation, and reasoning. However, the design and deployment of such modules in future urban scenarios is not trivial. In fact, engineering collaborative and coordinated components capable of harnessing the human and ICT capabilities in sensing at a large urban scale, shakes current approaches until their foundations rooted in top-down design. Designing with a top-down approach means that all the requirements of a software architecture have to be taken in account a priori; systems engineered in this way have a predictable and measurable behavior

but are not well suited to cope with dynamic execution contexts. Contrarily, bottom-up design delivers robust systems and can be fruitfully used in pervasive environments. However, predicting and controlling their behaviour *by design* is not an easy task.

Short coming urban scenarios call for a balanced trade off between the two approaches in designing an awareness framework. In fact, both top-down design and self-* properties are required. In this context, the contribution of this paper is twofold:

- it describes an awareness framework for knowledge collection using industrial-level tools. It is able to collect data from a number of different sources and classify them using general-purpose algorithms. The framework is higly dynamic; modules can be loaded, unloaded and reconfigured at runtime.
- it shows, using a case study, how state based automata could enable self-* properties and optimizations within the framework. More in detail, experimental results quantify their effectiveness in: *(i)* improving energy efficiency, *(ii)* improving classification accuracy and *(iii)* improving software engineering of aware systems.

The rest of the paper is organised as follows. Section 2 describes motivations and challenges behind this work. Section 3 presents the global architecture of our awareness framework, while a case study is sketched in Section 4. Preliminary experimental results have been explained in Section 5. Section 6 discusses related work. Section 7 draws conclusions.

2 Reconfigurable components and knowledge collection

Smart phones are increasingly equipped with computational, connectivity and sensing capabilities. At the same time, autonomous ICT devices (camera, drone, sensor networks and intelligent object) with sophisticated sensing capabilities are pervading our cities. In this scenario, entities with heterogeneous sensing capabilities are involved in complex sensing tasks.

However, this scenario implies novel challenges in knowledge collection: *(i)* different approaches and algorithms are needed to effectively deal with the many facets of real-world problems; *(ii)* the same classification scheme often requires to be tuned at runtime; *(iii)* classification accuracy is inversely proportional to the number of treated classes [18] [11].

Thus, modern architectures dealing with these requirements should exhibit a certain degree of dynamism and flexibility by making applications able to autonomously reconfigure. Service oriented and dynamically reconfigurable components have been proposed as a solution [15]. In particular, they can be deployed and removed at runtime. These features allow to select among different components depending on the situation. For example, given a specific classification task, it is possible to select a highly precise and computationally expensive algorithm or a less precise one considering the device. Furthermore, reconfigurable components can transparently modify their internal parameters. For example,

classifiers can analyse temporal windows of different sized based on the availably of computational resources, energy boundaries, or environmental conditions.

Despite reconfigurable components could provide a higher degree of flexibility to applications, the problem of driving reconfiguration processes is still open. In fact, every system capable of changing its internal structure or parameters must take decisions according to its environment. We decided to start driving internal reconfigurations with state-based automata because of their simplicity and generality. Each status is associated with a set of sensors specifically configured, while transitions and consequent reconfigurations are triggered by specific conditions. Moreover, such models are simple to understand and configure by developers.

Summarising, the key idea consists in using service oriented, reconfigurable components to classify data streams with data-driven models, while specification-driven models, such as automata, can drive the reconfiguration of the architecture. This design approach leads to three main advantages:

- *Improve energy efficiency.* In particular, for each specific situation the less energy demanding sensors and classifiers could be used. For example, it is possible to roughly recognise the vehicle used by a user with either GPS or accelerometer or microphone. An energy constrained system could constantly monitor its energy consumption and select the most appropriate trade-off. Furthermore, classification algorithms could be parameterized to be less precise in favour of a minor computational complexity.
- *Improve classification accuracy.* In the near future, the number of sensors available will rapidly increase and several classifiers could be used to recognise specific situations. It is up to the awareness framework to activate on-demand the most suitable sensors and classifiers by taking in consideration the current context awareness of the framework (e.g. the vehicle classifier could be activate when the speed overcomes a certain threshold).
- *Improve software engineering.* Organising the framework around the idea of reconfigurable components (i.e., sensors, classifiers) leads to modularity and composability of the software ecosystem. Users will be able to deploy components that are: *(i)* already included in the framework or *(ii)* developed by the developer community.

3 Internal architecture

This section details an awareness architecture supporting ideas described in Section 2. Specifically, its goal is to provide general-purpose awareness that could be used as a starting point for many diverse applications. Developers are only required to select the needed modules, define the topology of data flows and specify their reconfiguration strategies as depicted in Figure 1.

The architecture is structured around three layers, namely *sensor*, *classifier* and *awareness* layer. Each layer can host multiple modules connected each other via application-definable topologies. The data flow from sensors (i.e., both

hardware and software) through the whole architecture by means of in-memory queues enabling modules decoupling and many-to-many asynchronous communications. Each layer can host multiple modules (i.e, sensors, classifiers, awareness modules, queues).

The sensor layer hosts modules that are in charge of retrieving raw data from physical/social sensors and preprocess them. An example could be a module acquiring images from a camera and cropping and resizing them. Other examples could be modules acquiring facts from social networks, such as Twitter, Facebook or Foursquare. At the time of writing, we have already implemented modules for reading data from Android devices.

The classification layer hosts modules that consume data coming from the sensor layer and classify them (i.e., generate semantically richer information). An example could be a module able to classify the activity performed by a user by processing accelerometer data. At the time of writing, we have implemented modules for classifying user activity, location, speed, vehicle used on the basis on common smartphone sensors. Specific applications will need their own modules to be developed.

The awareness layer hosts modules consuming labels produced in the classification layer and feeding external applications with situational information. These modules might have different goals depending on the application. However, they could be divided into two main classes.

The former comprises modules delegated to sensor fusion processes. These modules receive labels, eventually conflicting, coming from multiple classification modules and apply algorithms to achieve higher semantical levels. For example, commonsense knowledge has been recently proposed [5] and could be integrated at this level.

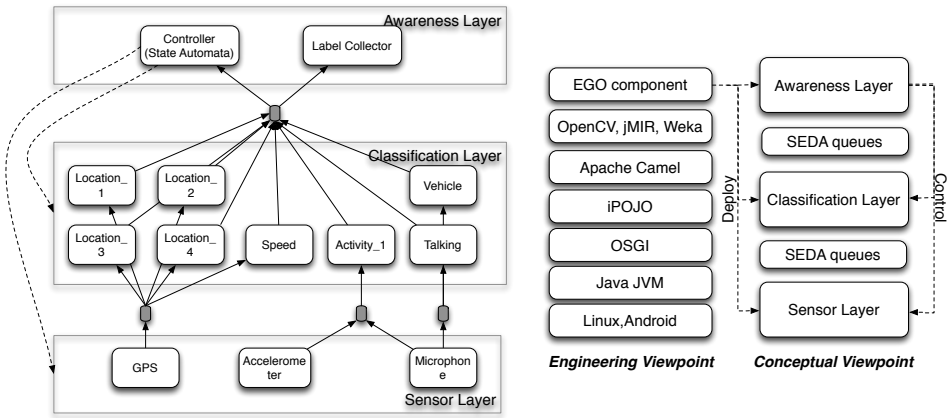

Fig. 1. Conceptual and engineering viewpoints of the architecture. The framework is structured around three layers, namely *sensor*, *classifier* and *awareness* layer and it is implemented on the top of industrial-level Java technologies.

The latter, instead, is related with the capability of the framework of monitoring and controlling itself. In a sense, the awareness layer could be the key of building a *self-aware* awareness module. For example, it would be possible to integrate within this level modules observing the internal status of the framework and activating different classifiers and sensors depending on operating conditions. This capability could be used to achieve both improved classification accuracies and reduced power consumption levels by continuously selecting to most suitable classifiers and sensors.

The strategies used to drive reconfiguration might vary depending on the application. For example, in Section 4 we propose a meta-classification scheme based on a simple automata in which each state in associated to a set of active sensors and transitions are triggered by their labels.

3.1 Implementation Insights

From an engineering viewpoint, the architecture is implemented on the top of industrial-level Java technologies. The reconfiguration mechanisms are provided by OSGi, a well-known Java framework that provides the typical features supported in a Service Oriented Component model.

On top of OSGi, we have an iPOJO layer. iPOJO is a container-based framework handling the lifecycle of *Plain Old Java Objects (POJOs)* and supporting management facilities like dynamic dependency handling, component reconfiguration, component factory, and introspection. Moreover, the iPOJO container is easily extensible and allows pluggable handlers, typically for the management of non-functional aspects. Each module deployed in the three layers is actually an iPOJO component able to meet the requirements mentioned in Section 2 [4].

The communications between layers is handled by exploiting a staged and even-driven approach in order to decompose a complex, application into a set of layers loosely connected. We build the support for the staged and layered architecture by making use of Apache Camel. This framework gives the capability to the iPOJO components in the different layers of asynchronously processing data streams and communicate through in-memory queues. These queues allow modules belonging to different layers to continuously communicate each other with minimum hardware requirements.

Finally, considering that pattern classification and analysis has a central role in situation awareness, we wrapped well-know data manipulation libraries within the framework such as Weka, jMIR and OpenCV.

4 Life logging application, a case study

To drive reconfiguration we started experimenting with state-based automata because of their simplicity and generality. A number of real-world problems can, in fact, be described using this approach and it is possible to drive the reconfiguration process using simple and clear schemas.

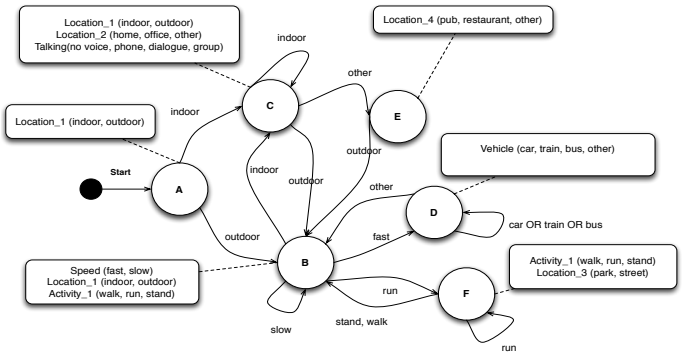

Fig. 2. Examples of state based-automata driving a self-aware smartphone collecting data about the life of the user.

To demonstrate that, in this section we describe a self-aware smartphone application collecting data about the life of the user. In particular: her activity, kind of location, vehicle used and people talking around.

Automata are defined by a list of its states and transitions between states. The current state in the automata defines active sensors and classifiers in the framework (Figure 2); it can change from one state to another when initiated by a triggering event, i.e a particular situational information or label produced by an active classifier; this is called a context-aware transition. A context-aware transition could triggers *(i)* an automatic reconfiguration or deploy of new sensors and classifiers; *(ii)* eventually an undeploy of active classifiers.

Thanks to this design approach modules can be enabled, disabled, wired and rewired in a dynamic way by making use of their output to navigate a state automata. For example, state C activates both the location and activity sensors but triggers transitions to itself on state B using only location labels.

The reconfiguration strategy driven by this automata improves *(i)* energy efficiency because costly component in terms of energy consumption are activated on-demand (e.g. when the system reaches state D, the only microphone is used to perceive changes); *(ii)* classification accuracy because classifiers work in optimal conditions (e.g. labels produced by the vehicle classifier are avoided when the user is located indoor). Furthermore, *(iii)* software engineering is greatly simplified. One can simply draw an automata, associate to each status sensors and classifiers and deploy the application.

5 Experimental evaluation

To assess our ideas we evaluated how *(i)* an automata-driven meta-classification scheme impacts on both classification accuracy and energy consumption; *(ii)* the framework behaves in terms of performace and scalability. We have tested the

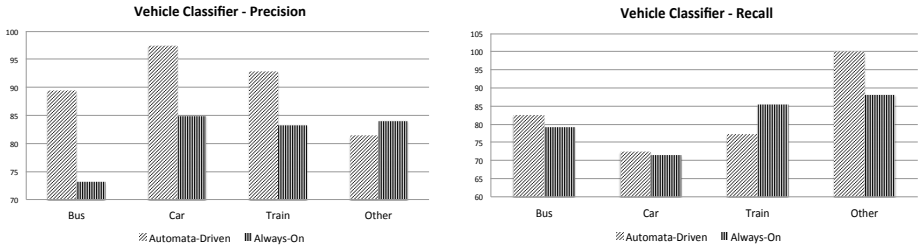

Fig. 3. Precision and recall of vehicle classifier obtained by classifying a dataset with or without automata-driven reconfiguration.

framework on a server with Core Duo 2 CPUs operating at 2.2GHz and 5GB of RAM. In the tests we used MacOSX 10.6 with JVM v1.6. The framework is running on Apache Karaf 2.3.0 OSGi container.

5.1 Smartphone case study

To evaluate the framework we implemented the smarthpone case study on top of our self-aware architecture. The dataset has been collected using an Android Galaxy Note II smartphone running a modified version of Funf [3]. Ground truth has been manually annotated. We defined a situation $s = \{activity, location, speed, vehicle\}$. Each field of the tuple can assume specific values. In particular $activity = \{walk, run, stand, sit\}$, $location = \{indoor, outdoor\}$, $speed = \{slow, fast\}$, $vehicle = \{car, bus, train, other\}$.

Each field of the tuple is managed by a specific classifier (see Figure 1-left). In particular: *(i)* an activity classifier using accelerometer and microphone data, that has a discriminative core based on SVM that uses a 64 dimensions Simple Magnitude Spectrum feature vector and its maximum value; *(ii)* a location classifier recognising indoor and outdoor environments when there is a lack of GPS data in a sliding window of 15 seconds; *(iii)* a speed classifier recognising fast and slow movements by computing the average speed over a sliding window of 10 GPS samples; *(iv)* a vehicle classifier using microphone data collected at 44.100Hz, 16bit, mono and divided in windows of 4 seconds long, that is based on SVMs and two feature vectors computed for each window using jMIR: a 13 dimension Mel-Frequency Cepstral Coefficient (MFCC) feature vector and a 10 dimension Linear Prediction Coefficients feature vector.

We recorded 7200 seconds of GPS, accelerometer and microphone data streams collected by three different users. The dataset has been divided into two equal non-overlapping sets, one used as training, the other as test set.

Results, summarised in Figure 3, show that classification precision for the vehicle classifier increases around 10% while recall does not vary significantly. The improvement derives from the fact that automata avoid all the errors produced by sensors working without context (e.g., a vehicle classifier working while users are walking in a busy street).

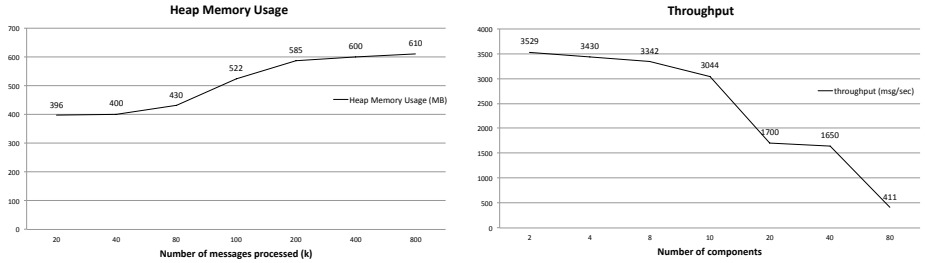

Fig. 4. Evaluation of memory consumption and throughput of the framework by increasing the number of parallel components in the classifier layer.

Furthermore, these results have been achieved with a substantial reduction of the energy required because the GPS is active around 46% of time, the microphone 54% and the accelerometer 6%, instead of being always-on.

5.2 Performance evaluation

As experimental evaluation, we have estimated the performance of the framework by dynamically increasing the number of parallel components in the classifier layer that process a data stream of 10k messages (injected as fast as possible and with small payload) without further computations. The effect is an increasing number of messages that have to be processed in the framework. The corresponding experimental results are reported in Figure 4

The first metric used to evaluate the performance is the memory usage. In particular, the graph shows how the heap memory usage increases linearly with the increasing number of messages processed.

The second metric is the throughput, i.e. average number of messages per second processed by all the components in the classifier layer. In particular, the graph shows how the average throughput of classifiers decreases linearly with the increasing number of components consuming the data stream. However, in the worst case (eighty parallel components) the average throughput remain acceptable (411 msg/sec).

These results demonstrate how *(i)* the bottleneck in the awareness module is the CPU that limits the scalability in terms of number of messages processed per second; *(ii)* the overhead introduced by our awareness module is negligible, because the performance decreases linearly with an increasing number of components.

6 Related work

In the last years, researchers prototyped sensing systems able to acquire detailed situational information from data streams [6, 13, 12] using both specification-driven (e.g., logic ontologies, logic programming, fuzzy logic) or data-driven ap-

proaches (e.g., support vector machine, decision trees, neural networks). The most prominent works have been surveyed in [18] and [11].

However, the majority of these systems lacks in generality and addresses specific classification problems, by making use of a pre-defined set of sensors [14]. Few of them tried to make use of both the approaches designing frameworks that are resource efficient and robust in a large plethora of situations. For example, in [15] authors make use of processing pipelines on various sensors (i.e., GPS, accelerometer) to show how dynamic reconfiguration could be used in continuous sensing. However, the framework doesn't use any specification-driven approach.

On the other hand, Cimino et. al. [7] describe a framework using specification-driven reasoning to deal with heterogeneous user behaviours. However, it doesn't allow awareness to drive the reconfiguration of sensors and classifiers. Other works [16], [8] propose to optimise the sensing process in terms of power saving. In particular, the former [16] exploits a specification-driven reasoning technique to learn relationship among context attributes to optimise the internal logic of an awareness framework. However, like previous works, these frameworks focus only the optimisation of energy consumption in continuous sensing.

Kawsar et. al. [9] demonstrate that our approach of modelling human activities and contexts with a set of states glued by a set of transitions can successfully describe a number of real world scenarios and drive the reconfiguration process.

To the best of our knowledge, our self-aware framework represents a first attempt to integrate the two approaches in a more general way. The framework is able *(i)* to make full use of all the strategies proposed in previous works and *(ii)* to deal with more complex scenarios rooted in self-* requirements.

7 Conclusion and future work

In this paper we proposed an innovative awareness framework suitable for developing pervasive applications. It has been designed around the concept of reconfiguration and built using industrial-level tools. Its modular and portable architecture has been provided with a meta-classification scheme based on state automata to drive reconfiguration, with promising results in terms of both classification accuracy and energy consumption.

As future work, we plan to challenge our reconfigurable framework in more complex scenarios to better understand how the framework self-* structure could simplify the engineering of a mobile context-aware operating system.

Acknowledgments: work supported by the ASCENS project (EU FP7-FET, Contract No. 257414).

References

1. *Smart cities Ranking of European medium-sized cities.* http://tinyurl.com/bqh83np, Vienna, Austria, 2007.

2. Looking ahead in pervasive computing: Challenges and opportunities in the era of cyber physical convergence. *Pervasive and Mobile Computing*, 8(1):2 – 21, 2012.
3. N. Aharony, W. Pan, C. Ip, I. Khayal, and A. Pentland. Social fmri: Investigating and shaping social mechanisms in the real world. *Pervasive and Mobile Computing*, 7(6):643–659, 2011.
4. P. Bellavista, A. Corradi, D. Fontana, and S. Monti. Off-the-shelf ready to go middleware for self-reconfiguring and self- optimizing ubiquitous computing applications. *Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication*, 2011.
5. N. Bicocchi, M. Lasagni, and F. Zambonelli. Bridging vision and commonsense for multimodal situation recognition in pervasive systems. In *International Conference on Pervasive Computing and Communications*, Lugano, Switzerland, 2012.
6. D. Choujaa and N. Dulay. Tracme: Temporal activity recognition using mobile phone data. In *Embedded and Ubiquitous Computing, 2008. EUC '08. IEEE/IFIP International Conference on*, volume 1, pages 119–126, 2008.
7. M. Cimino, B. Lazzerini, F. Marcelloni, and A. Ciaramella. An adaptive rule-base approach for managing situation-awareness. *Expert System with Applications*, 39(12):10796–10811, 2012.
8. S. Kang, J. Lee, H. Jang, Y. Lee, S. Park, and J. Song. A scalable and energy-efficient context monitoring framework for mobile personal sensor networks. *Mobile Computing, IEEE Transactions on*, 9(5):686–702, 2010.
9. F. Kawsar, G. Kortuem, and B. Altakrouri. Designing pervasive interactions for ambient guidance with situated flows. 3:371–375, 2010.
10. M. Kehoe and al. Understanding ibm smart cities. *Redbook Series*, 2011.
11. W. Z. Khan, Y. Xiang, M. Y. Aalsalem, and Q. Arshad. Mobile phone sensing systems: A survey. *IEEE Communication Survey and Tutorials*, 15:402 – 427, 2013.
12. J. R. Kwapisz, G. M. Weiss, and S. A. Moore. Activity recognition using cell phone accelerometers. *SIGKDD Explor. Newsl.*, 12(2):74–82, Mar. 2011.
13. K. Lin, A. Kansal, D. Lymberopoulos, and F. Zhao. Energy-accuracy trade-off for continuous mobile device location. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, pages 285–298, New York, NY, USA, 2010. ACM.
14. H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell. Soundsense: scalable sound sensing for people-centric applications on mobile phones. *Proceedings of the 7th international conference on Mobile systems, applications, and services,*, pages 165–178, 2009.
15. H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell. The jigsaw continuous sensing engine for mobile phone applications. *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, pages 71–84, 2010.
16. S. Nath. Ace: exploiting correlation for energy-efficient and continuous context sensing. pages 29–42, 2012.
17. H. Schaffers, N. Komninos, M. Pallot, B. Trousse, M. Nilsson, and A. Oliveira. The future internet. chapter Smart cities and the future internet: towards cooperation frameworks for open innovation, pages 431–446. Springer-Verlag, Berlin, Heidelberg, 2011.
18. J. Ye, S. Dobson, and S. McKeever. Situation identification techniques in pervasive computing: A review. *Pervasive and Mobile Computing*, 8:33–66, 2012.
19. F. Zambonelli. Toward sociotechnical urban superorganisms. *IEEE Computer*, 45(8):76–78, 2012.