

Towards behaviour inference in smart environments

Mário Antunes

Diogo Gomes

Rui Aguiar

Instituto de Telecomunicações, Universidade de Aveiro, Aveiro, Portugal
{mario.antunes, dgomes, ruilaa}@av.it.pt

Abstract—Smart environments are physical places that are richly and invisibly populated with sensors, actuators and computational elements. The objective of such environments is to adapt themselves to its users in order to increase their comfort and usefulness. This paper proposes a platform, named APOLLO, capable of inferring behaviour rules from a smart environment and apply them to provide an intelligent space. The APOLLO platform is built upon a Service Oriented Architecture (SOA), in which collected context information is used to infer behaviour rules through statistical and machine learning techniques. The proposed platform is to be deployed in a home automation scenario.

Keywords—Smart environments, knowledge extraction, machine learning, SOA

I. INTRODUCTION

Smart houses, smart environments and ambient intelligence have been important research topics since 1940. In that year Chevrolet produced a commercial video named “Leave it to Roll-Oh”, where a robot served as the family butler. The video ended pointing out that a robotic butler was still a dream while presenting several small machines that helped in daily activities, e.g. electric coffee machine, luminosity sensor, automatic water sprinkler. This video shows how important the concept of smart environments became to the industry.

Nigel Shadbolt [1] identifies three computational areas that must converge in order to develop a truly smart environment. The first computational area is ubiquitous or pervasive computing, responsible for providing a seamless interface between the environment and its users. The system must be integrated with everyday use objects, allowing a natural and simple interaction between it and the users. The second computational area is intelligent systems, responsible for inferring the context of the environment and learn useful patterns based on the users behaviour. This can be achieved through several techniques e.g. data mining, statistical analysis, machine learning and optimization methods. The third and final computational area is context awareness: in order for the system to adapt to the user’s habits, it needs to perceive the context of the environment (sensors) and how to change it (actuators).

This paper presents a platform, part of the APOLLO project, that is capable of inferring behaviour rules from a smart environment. The platform collects raw data from sensors that are scattered through the environment, processes the data and infers behaviour rules from it. These rules depend on the type of data collected and on the type of environment. The platform is able to detect what patterns are relevant and how to enforce them through actuators.

The APOLLO project’s main objective is the development of a platform that supports new services in the area of machine-to-machine (M2M) communications. The project aims to develop a transversal technological platform that supports management, control and monitoring of an heterogeneous network of sensors and actuators. APOLLO exports a service layer to third parties willing to develop next generation M2M applications in various areas including Utilities, Transports, Health, Agriculture, Distribution and Consumer Electronics. The project platform will support from its start a vast set of M2M Smart Services & Applications such as Smart Metering, Smart Grids, m-Health (remote monitoring of patients), Smart Cities, Smart Home and Smart Buildings according to a Portuguese Government policy for the deployment of next generation networks. This paper will focus on the aspects and services related to inference of rules based on the data collected by the platform, and on the supply of rules to the same platform.

The remainder of the paper is organized as follows. Section II presents and discusses the proposed platform. Section III discusses and details how the platform can be used to optimize the comfort of inhabitants in a smart home. Section IV discusses the related work. Finally the conclusions and the future work are presented in Section V.

II. APOLLO: CONTEXT-AWARE RULE INFERENCE

In this paper we propose a novel platform that infers behaviour rules from a smart environment. The inferred rules details relevant knowledge about the environment. As an example, in a home automation scenario, the proposed platform is capable of automatically adapting to the inhabitants habits.

Our solution intends to be a platform capable of correctly inferring behaviour rules from an open scenario. Currently similar systems rely in a controlled environment (close scenario), a environment where all the sensors and actuators are completely detailed. We intend to develop a platform that is independent from the underlying scenario. Independence from the underlying scenario can be achieved if it becomes possible to add or remove any type of sensor or actuator without the need of manually reconfiguring the platform reasoning engines. This is an important advantage since these types of equipments are highly heterogeneous and tend to evolve rapidly.

The APOLLO platform is divided into several independent services, following a Service Oriented Architecture (SOA). This type of architecture has three important advantages. First it is trivial to update any service in the architecture, since each

service is independent of each other. Second the communication of the various services is based on message passing patterns, and because of this the complete system becomes highly parallelizable. This means that the system takes advantage of all the processing resources available, such as those found in modern multi-core processors or cloud computing offerings. Third, this approach provides a competitive ecosystem where optimum products can be independently developed.

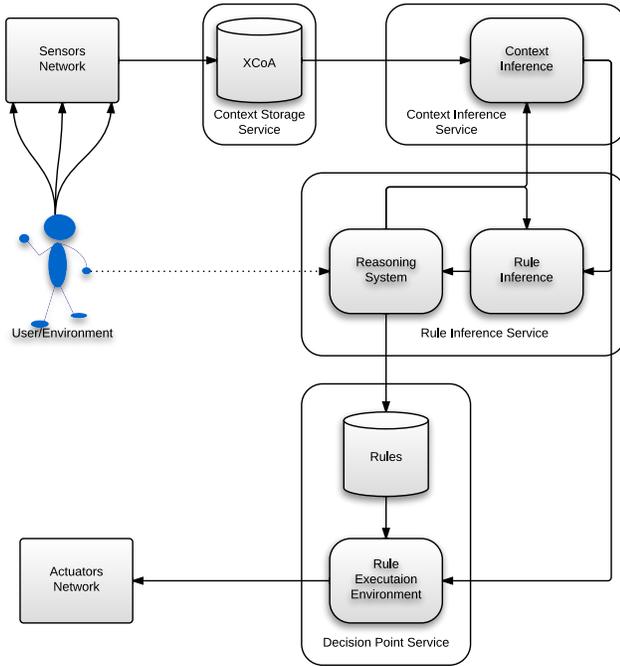


Fig. 1. Architecture and information flow between the several services of the proposed platform.

Fig.1 shows the architecture and the information flow between the several services of the platform. As previously referred, one of the objectives of the platform is to be independent from the underlying environment. The first step to achieve this objective is to be as independent as possible from the sensors scattered through the smart environment. It is possible to achieve this through two different services: Context Storage and Context Inference. The Context Storage service uses a unstructured storage system in order to accept any type of document, and therefore is not limited to previously provisioned data types. The Context Inference service, is connected to the context storage service, receives raw data from the sensors inferring the context information from it and creating a logical structured element. These elements allow the remaining services to perceive a uniform structure of information regardless of the smart environment and underlying sensors.

The Rule Inference service is composed by two distinct components. The first component, designated Rule Inference Component, receives the context, inferred by the previous service (Context Inference service), and through several data mining, statistics and machine learning techniques, detects relevant patterns and infers behaviour rules that can be observed in the environment.

The second component, designated Reasoning System,

receives the behaviour rules inferred by the rule inference component and verifies for logic correction and usefulness. The logic correction is automatically verified by the service while, at this point in the development, the usefulness of the rule may be verified by a human user. This service offers a graphical user interface, through which a human can decide what rules are useful or not. After some time (training period) the system learns the users preference as a consequence manual corrections will be less frequent.

The rules that are considered correct and useful are sent to the decision service. Based on the context of the environment and on the current set of rules, this service performs several tasks, activating or de-activating actuators present in the environment.

In the rest of the paper the Context Inference and Rule Inference services will be designated as the Inference Pipeline. In the following subsections each service that composes the platform is discussed in greater detail.

A. Context Storage service

One objective of the APOLLO platform is to be as independent as possible from the underlying environment. To achieve independence from the underlying environment it is necessary to develop a context storage service that accepts any type of information. Relational databases are not suited for this task. As the name implies, a relational database relies on relations defined in advance by a human. Due to the heterogeneity of potential relevant information it is not possible to define in advance all the possible relations. Depending on the smart environment it might not even be possible to establish all the relations between the sources of information. The relations are also likely to evolve through time with the evolution of the smart environment and the addition or removal of some of its sensors and actuators.

To tackle this problem it was adopted a solution based on a NoSQL (Not only SQL) database [2], as proposed by Nuno Santos et al. [3]. NoSQL databases do not require the definition of relations between the structure of the stored information, hence this type of databases are able to store any document.

How the devices connect to the context storage is another important issue, it is necessary a uniform interface that any device can use in order to publish/retrieve context information. Currently there are two major types of context storage architectures. The first architectures are purely based in middleware, e.g. EU-funded Music project[4]. These have the disadvantage of not being able to address the heterogeneity of the underlying devices. Other architectures are web base, e.g. EU-funded Mobilife project [5]. These architectures, on the other hand, are unable to cope with the real-time and reactivity that many context aware services require. The architecture of the context storage service adopted was proposed by Diogo Gomes et al. [6]. It is based in XMPP (Extensible Messaging and Presence Protocol) protocol and supports the publish-subscribe model.

B. Context inference service

The main function of this service is to gather information from the sensors and organize it into logic datasets that can

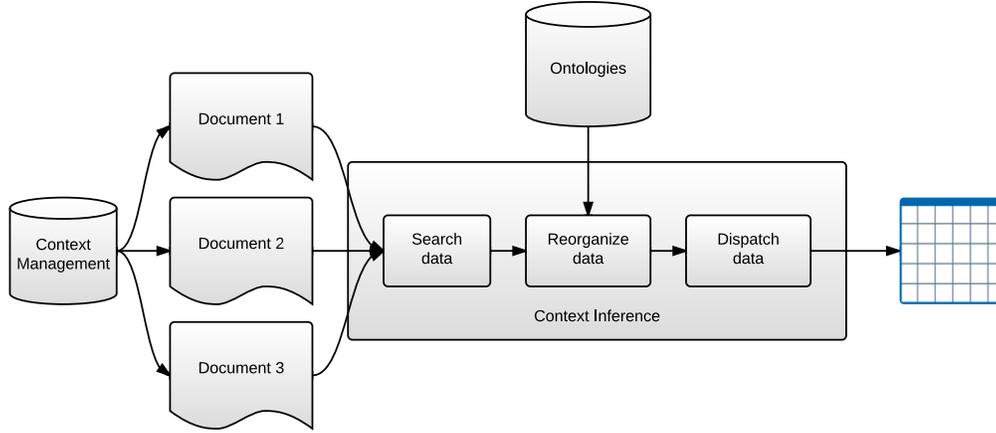


Fig. 2. Information flow of the context inference service.

be analysed by the other components of the platform. Fig.2 shows the information flow of the context inference service.

The context inference service queries the context storage service for documents containing raw data from the sensors scattered through the environment. To access information the service only needs to subscribe the nodes correspondent to the relevant sensors.

The data collected from the sensors is chronologically organized and stored into a dataset. Due to the continuous increase of information (added by the sensors) the dataset is perceived as a continuous stream. The context inference service divides the stream into independent and tractable datasets through a sliding window. The service is able to cope with the variability of the data and automatically adjusts the sliding window (for example using an adaptive window [7]).

C. Rule inference service

The Rule Inference service is divided into two different components:

1) *Rule inference component*: The rule inference component analyses the dataset received from the context inference service. Each dataset is independent from the others, hence it is possible to process several datasets in parallel. The first step of the rule inference component is to analyse the dataset using statistical and machine learning methods. Through this process it is possible to detect relevant patterns. From the relevant patterns, simple behaviour rules for the environment are then inferred.

The second step of the rule inference process is the optimization of the rule previously inferred. Using the dataset, the service trains a predictive model. This model allows the service to predict the behaviour of the smart environment with a different combination of inputs. The regression model is then used in the optimization step as it allows to predict the environment behaviour. Currently, a genetic algorithm [8] has been implemented as the optimization algorithm. Depending on the scenario, a defined objective function will be used by the optimization algorithm. It is important to notice that at this point it is not required a highly precise model, since the platform can verify the results of the rules it infers.

2) *Reasoning component*: This component has two main functionalities. First, and since the rules are automatically optimized, it verifies if the inferred behaviour rules are useful to the users of the environment. In order to validate the usefulness of the rules there is often the need of validation by a human user. This is achieved with an interface that allows the inhabitants to monitor and decide which rules are useful to them.

Second, and based on the validation from the users, it verifies the quality of the inferred rules. Using this information the service sends messages to the context inference service in order to adapt the context inference process. As an example, the service can control the size of the sliding windows through the stream of data.

D. Decision service

The decision service receives the behaviour rules from the inference service and stores it into a database. Based on the context of the environment and the current set of behaviour rules, the decision service activates or deactivates several underlying actuators. The addition or removal of behaviour rules, depending in the environment, can be very dynamic. As an example in a home automation system the evaluation of behaviour rules can be associated with seasonality. However for other scenarios, such as critical systems, the behaviour rules can change more rapidly.

This service is also responsible for abstracting the underlying actuators in the system, similar to the abstraction of sensors achieved by the context storage and context inference services.

III. HOME COMFORT/ENERGY OPTIMIZATION

We instantiated the proposed platform in a home automation scenario. There are already several commercial smart home solutions [9]–[11], which cover the first and the third computational areas described in Section I (ubiquitous computing and context awareness). Usually the second computational area, intelligent system, is oversimplified by manufacturers using a fixed set of rules and parameters specified by the inhabitants of the environment. This simple approach hinders these home solutions, as most users are either not skilled

enough to configure the system or uninterested due to the inflexibility of the system.

Current commercial systems reacts based on a specified set of rules, performing several quotidian tasks individually without flexibility to cope with variations of the inhabitants behaviours (e.g. vacations, family events, etc). The most common ones are opening/closing the window curtains, manage temperatures and monitor the exterior of the environment for security reasons. These systems are not able to learn the inhabitants habits and adapt to it. Current home automation systems cannot be considered smart environments, since they lack the ability to automatically adapt to the inhabitants needs. In commercial systems the adaptation is only achieved if the inhabitants define a new set of rules each time their behaviour evolves. For example, current commercial solutions cannot be utilized to monitor and help the elderly persons. It is unrealistic to expect that an elder person will define new rules, behaviours or patterns as necessities evolve. On the other hand a truly smart environment can be used by these persons in order to achieve greater independence and quality of lifetime.

The following Subsection III-A presents the designed scenario. Subsection III-B details the implementation of the platform for the devised scenario, and Subsection III-C presents the results of the evaluation.

A. Scenario

For the purpose of evaluating the platform, we have considered a reference scenario in which the platform receives temperature and energy consumption data, and define the temperature of the air conditioning system, as depicted in Fig.3. The objective is to learn the temperature preferences of the inhabitants, while optimizing the energy consumption.



Fig. 3. Designed test scenario for this project. The scenario is composed of two sensors (temperature and energy sensor) and one actuator (air conditioning system).

At this point in the project the physical smart environment is still being built. The test alternative was to use a dataset from another smart house research project that has similar sensors. The platform was tested using a dataset publicly available at [12]. The data was collected from sensors in a smart house test-bed from Washington State University (WSU), during the Summer of 2009. In this scenario, with minimal intervention from the users, the system should detect the preferred temperature and optimize the energy consumption required to achieve it.

B. Implementation

In order to adapt our platform to this practical scenario it was required the customization of two services, the context inference and the rule inference system. The customizations are discussed as follows.

1) *Context inference*: As previously referred, the context storage service accepts any document from the sensors. The context inference service analyses each document, extracts the relevant information and organizes it into a stream of data.

For this implementation it was used a database of ontologies that expresses what variable depends on each other. The dataset extracted from the stream was organized in columns as follows:

- Energy spent by the air conditioning system
- Temperature measured in the environment
- Time at which the previous values were read

A method to automatically organize variables collected from sensors without a ontology database is currently being defined, and will be addressed in future publications. Several sizes of sliding windows were used, with results presented in Section III-C.

2) *Rule Inference*: The analysis of the dataset is performed as follows. The dataset is statistically analysed in order to discover the most frequent temperature and what time of the day that temperature was measured. A histogram of temperatures is built, and the most frequent temperature is selected as the target temperature of the behaviour rule. It is also built an histogram that contains the time periods when the most frequent temperature was read by the sensor in the environment. The most frequent time period is selected as the target period of the behaviour rule. At this point the platform generates a simple behaviour rule based on the most frequent behaviour of the users.

A Support Vector Machine (SVM) [13] regression model was trained. The model allows the system to estimate the energy consumption based on the temperature and time period. A genetic algorithm [8] was then used to optimize the detected pattern with the objective of lowering the power consumption. The optimization algorithm tries to achieve the target temperature in the target time period consuming the least energy possible. The regression model, computed through the dataset using a SVM, is then used in the optimization as it allows to compute the energy spent.

This optimization tries to verify if a slight decrease of temperature or a smaller amount of time that is unperceived by the users produces a meaningful result and savings in terms of energy. Is is assumed that the platform is able to automatically generate objective functions from the inferred rules.

All optimization algorithms need an objective function in order to compare how good the optimizations are compared to the initial solution. In a genetic algorithm this objective function is designated as fitness function. Based on target temperature, target time and average energy spent to maintain that temperature within that time period, a fitness function is defined.

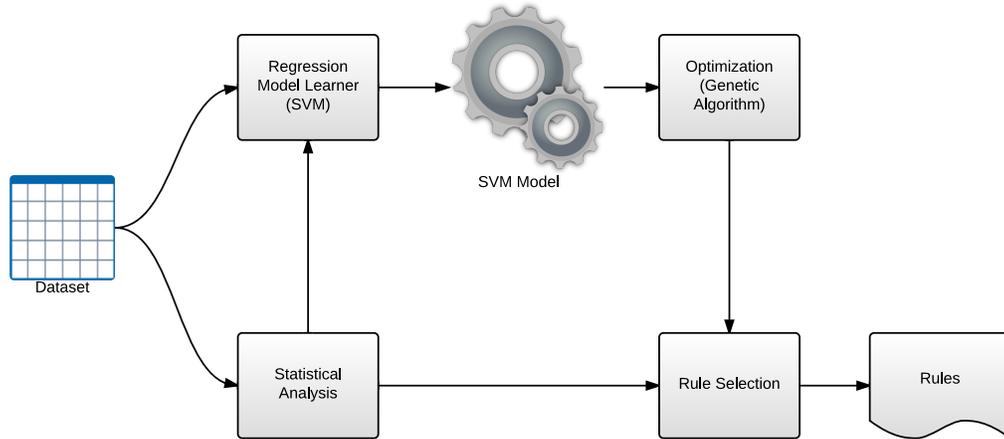


Fig. 4. Architecture of the behaviour rule inference model.

Equation 1 presents the fitness function used. The function is divided into three components: energy fitness (EF), temperature fitness (TF) and time period fitness (PF). The time period and temperature fitness formulas tend to 1 when the optimized values tend to the target values. On the other hand, the energy fitness formula tend to 1 when the optimized value tend to 0. We wanted to obtain the target temperature at the target period consuming the least amount of energy. Each component has a weight associated (w_E , w_T and w_P respectively), that controls the effect of the respective component in the result of the optimization. For example, a high weight in the energy fitness and the optimized rules will allow to consume less energy at the expense of not reaching the desired temperature at the desired time period.

$$\text{Fitness} = \frac{w_E \times \text{E. F.} + w_T \times \text{T. F.} + w_P \times \text{P. F.}}{w_E + w_T + w_P} \quad (1)$$

$$\text{E. F.} = \frac{\text{average Energy} - \text{Energy}}{\text{average Energy}} \quad (2)$$

$$\text{T. F.} = \frac{1.0}{(\text{target Temperature} - \text{Temperature}) + 1.0} \quad (3)$$

$$\text{P. F.} = \frac{1.0}{(\text{target Period} - \text{Period}) + 1.0} \quad (4)$$

Finally all rules inferred are sent to the reasoning component, since the datasets are extracted from a possible continuous stream of data, it is inferred one rule from each dataset. The reasoning component generates a histogram and stores the rules inferred by the rule inference component. The rule with higher count in the histogram is the rule with higher probability of being the correct rule. Again since the stream is possibly infinite the system should converge to the rules that model the environment. The database of the decision point service is updated each time the most probable rule changes.

Fig.4 shows the architecture of the behaviour rule inference model.

C. Performance Evaluation

The APOLLO platform was evaluated taking into account the scenario described previously (Section III-A). It is relevant

to mention two important aspects. First, there are no physical sensors sending data for the context storage service, as they are still being built. Future publications will evaluate the performance of the platform using data gathered from physical sensors. Second, since the stream of data is static and there are no actuators to activate, it is not possible to verify with user interactions the effect of the inferred behaviour rules. Nevertheless the throughput of the system and the ability to detect relevant patterns were numerically evaluated.

The stream of data was manually stored in the context storage. The context inference service subscribes the context storage for documents and receives the complete stream containing three months of data. The rules inferred were analysed by some human users but it is not possible to verify if the rules correspond to the actual needs of the original human inhabitants.

We performed two different evaluations in the platform: first we analysed the throughput of the platform, second we analyse the ability to infer behaviour rules. The platform was deployed into a two separate servers, the context storage service was deploy in one server, with a 2.6 GHz CPU and 1 GB of memory RAM. While the reaming services were deployed into a more powerful server, with 24 CPUs and 200 GB of memory RAM.

The first performance evaluation was to measure the throughput of the system. For this evaluation it was simulated 75, 125, 250 and 500 simultaneous streams of data. Conceptually each stream represents a individual home. The generation and sending of a new dataset followed an exponential distribution with mean equal to 10 seconds. Each stream was divided into datasets containing data from 7, 15 and 30 days, which correspond to a week, two weeks and a month, respectively. For the three dataset sizes the sliding window always slides one day for each created dataset. We assume, for this practical scenario, that one day slide is a good compromise between reactiveness and new information for the system to infer new rules.

We measured the average time required by a dataset to be processed by the Inference Pipeline. The dataset processing time is measured from the moment the dataset is extracted

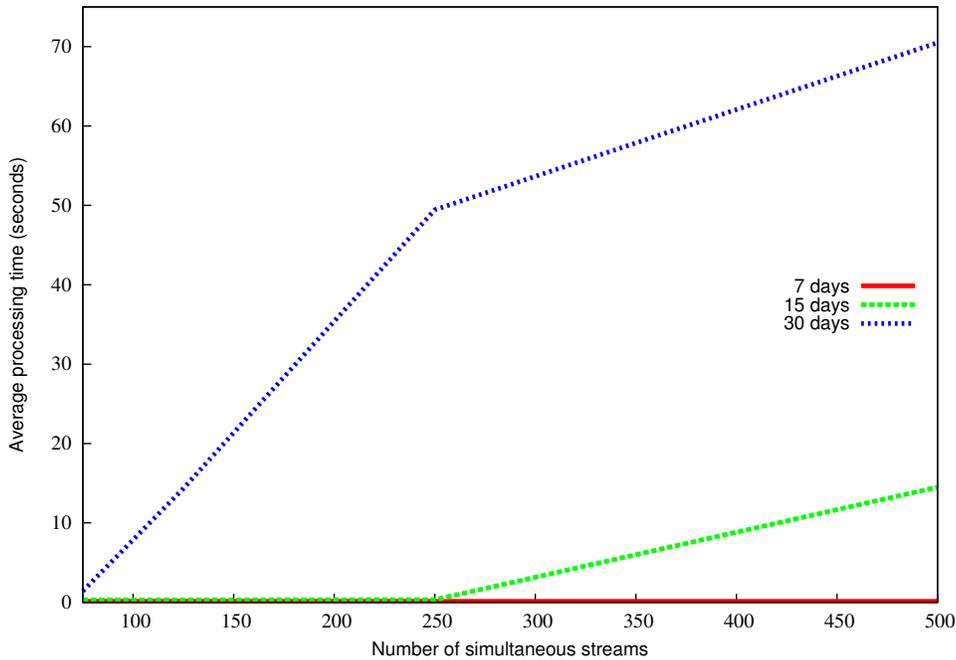


Fig. 5. Dataset average processing time. The x axis contains the number of simultaneous streams, the y axis contains the processing time in seconds.

until the the moment the rules are inserted in the rules database. Fig.5 shows the dataset average processing time for different dataset sizes and different number of simultaneous streams.

Even in the worst-case scenario (500 simultaneous streams and dataset with 30 days) it only takes 70 seconds in average to completely process a dataset. This processing time is completely satisfactory for the considered home automation scenario and to almost all home automation scenarios. If a specific scenario requires a higher throughput is possible to instantiate multiple context inference and reasoning inference services and divide the workload through them. This is possible because the platform follows a SOA and each service is as independent as possible from each other.

The second performance evaluation was the ability to detect relevant patterns. Image 6 shows a graphical representation of the final histogram obtained after analysing the complete stream with a sliding window of 7 days. The rule with the higher number of votes has almost double the votes than the second one. Another important aspect is that the most probable rules agree that the most relevant temperature is 25°C only differing in the time period. So it is possible to state that the system is capturing a relevant pattern, at least from a statistic point of view.

IV. STATE OF THE ART

Over the years, there have been several attempts to create fully autonomous and pervasive home environments.

Mozer [14], [15] applied neural networks [16] in home automation. The author developed a system that was able to control air, heating, lighting, ventilation and water heating. The main goal of the project was to anticipate inhabitants needs and conserve energy at the same time. For that, it applied reinforcement learning [16]. During a learning period,

inhabitants indicated their preferences whenever predictions were incorrect by selecting themselves what they would expect the system to do (for instance, if they want the lights on, and the system did not turn them, users simply turned them on). This way, the system would adjust itself to its inhabitants preferences.

Vainio et al [17] proposed home-control system that uses fuzzy logic rules [18]. Initial rules were given manually and, through reinforcement learning, the home adapted by replacing old rules with new ones. This method was applied to control a lightning system in a smart-home laboratory environment. The author concluded that users did not care if the rules generated by the system didn't match exactly what they wanted. It also concluded that, after a certain amount of time, if a rule had a big weight, it was likely to keep its importance, and new rules, that correspond to sporadic actions, were quickly eliminated.

The architecture proposed by Mozer [14], [15] depends on the type of environment, not being able to cope with the addition of new sensors nor actuators. The proposed platform adapts automatically, without making assumptions about the environment, and starts building environment rules from the raw data. *A priori* knowledge of the environment is not required, since the proposed platform is independent from the underlying environment. Another issue with the architecture proposed by Mozer is that the system does not infer the inhabitants preferences, requires a training phase where the inhabitant needs to specify his preferences over the preferences assumed by the system. Vainio et al [17] also proposed a proactive architecture, that requires a initial set of rules defined by the inhabitants. One advantage of the APOLLO platform is the fact that it does not need a training dataset in order to adapt to the environment. At first the inferred rules might not be that useful but with the increase of processed data the system converges to the set of rules that defines the expected

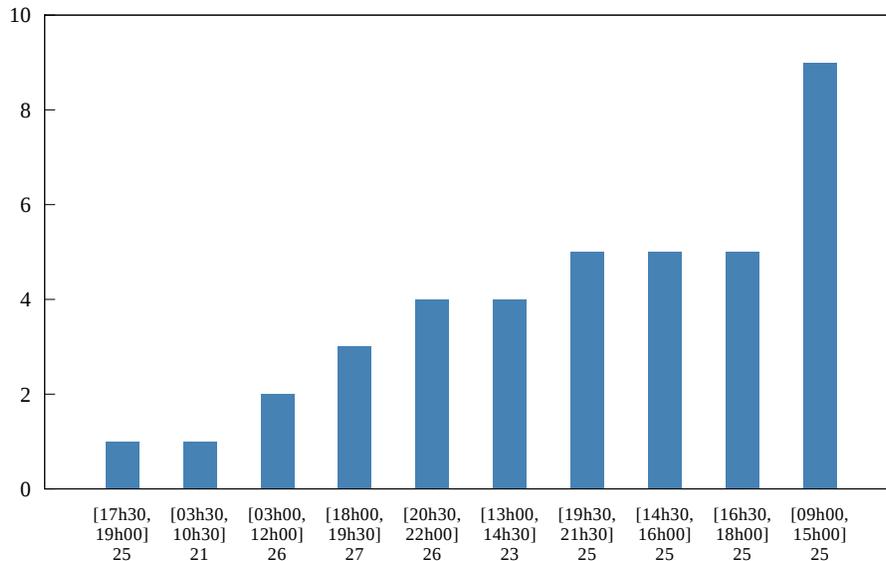


Fig. 6. Histogram of the rules inferred by the rule inference component. The x axis contains the inferred rules, the time period is between square brackets followed by the inferred temperature.

behaviour of the environment.

The smart thermostat [19] is an approach that detects occupancy and sleep patterns in a house in order to save energy. To save energy, it automatically turns off the heating, ventilation and cooling systems whenever inhabitants are sleeping or are not at home. The system uses historical data and on-line sensor data to decide if it should preheat the house or heat it only after it is occupied.

AIM project [20] aims to profile and reduce home energy consumption. It intends to predict user preferences through the monitoring of its behaviour and environmental parameters, such as user presence, temperature and light. Daily profiles are clustered through a cross-correlation between each couple of daily data. Also, the system uses real time data to dynamically update its predictions.

Ubiquitous Smart Energy Management (USEM) [21] is also a system to manage power usage. This system allows real-time monitoring of electricity consumption. Users can interact with the system in order to set their own rules and monitor appliances usage. The main improvement of this system is that it considers energy availability and price.

Much of smart environment research [19]–[21] is mainly focused in saving energy. These types of systems are not suitable for elder persons or persons with dementia.

V. CONCLUSIONS

This paper presents a new platform that is capable of automatically inferring behaviour rules from a smart environment. The APOLLO platform presents a new architecture that is as independent as possible from the underlying smart environment.

The most important part of the platform is the Inference Pipeline. The first service, designated context inference, processes the raw data from the sensors and infers the context

of the environment. The smart environment context is the necessary information to describe the environment state.

The second service, composed by two components: rule inference component and reasoning component. The rule inference component receives the environment context and through data mining, statistical analyses and machine learning techniques detects relevant behaviour patterns. Based on the detected patterns it infers behaviour rules. The reasoning component is responsible for verifying the quality of the inferred behaviour rules. Based on that, this component reconfigures the rule inference component and the context inference service. It is also responsible to verify the logic correction and usefulness of the inferred behaviour rules.

As previously stated, the APOLLO platform presented in this paper is an ongoing project. For this paper the implementation of the platform has adapted for the available dataset on home automation.

Areas of interest that will be analysed in the future with more detail are devising a technique that allows the system to automatically select weights in the fitness functions which will allow the platform to better adapt to the inhabitants needs and habits. Rule execution service and rule storage services are also areas of interest since the platform needs the ability to execute the rules inferred.

VI. ACKNOWLEDGEMENT

This work has been partially funded by the Portuguese Innovation Agency/National Strategic Reference Framework (AdI/QREN) under grant agreement No. 2011/021580 (APOLLO project).

REFERENCES

- [1] N. Shadbolt, "Ambient intelligence," *IEEE Intelligent Systems*, vol. 18, no. 4, pp. 2–3, Jul. 2003. [Online]. Available: <http://dl.acm.org/citation.cfm?id=939826.939889>

- [2] N. Leavitt, "Will nosql databases live up to their promise?" *Computer*, vol. 43, no. 2, pp. 12–14, February 2010.
- [3] N. Santos, O. Pereira, and D. Gomes, "Context storage using nosql," in *Conferência sobre Redes de Computadores*, 2011. [Online]. Available: <http://atnog.av.it.pt/publications/context-storage-using-nosql>
- [4] N. Paspallis, R. Rouvoy, P. Barone, G. A. Papadopoulos, F. Eliassen, and A. Mamelli, "A pluggable and reconfigurable architecture for a context-aware enabling middleware system," in *Proceedings of the OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008. Part I on On the Move to Meaningful Internet Systems.*, ser. OTM '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 553–570.
- [5] P. Floren, M. Przybilski, P. Nurmi, J. Koolwaaij, A. Tarlano, M. Wagner, M. Luther, F. Bataille, M. Boussard, B. Mrohs, and S. Lau, "Towards a context management framework for mobilife," in *In IST Mobile & Wireless Communications Summit*, 2005.
- [6] D. Gomes, J. M. Goncalves, R. O. Santos, and R. Aguiar, "Xmpp based context management architecture," in *2010 IEEE Globecom Workshops*. Miami, Florida, USA: IEEE, December 2010, pp. 1372–1377. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5700163>
- [7] A. Bifet and R. Gavald, "Learning from time-changing data with adaptive windowing," in *In SIAM International Conference on Data Mining*, 2007.
- [8] W. Banzhaf, F. D. Francone, R. E. Keller, and P. Nordin, *Genetic programming: an introduction*, M. K. P. Inc., Ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998.
- [9] L. Green, "Smart home systems," 2012. [Online]. Available: <http://www.limegreen.tv/residential/smart-home-systems/>
- [10] T. B. Group, "Your smart home," 2012. [Online]. Available: <http://www.thebelmontgroup.co.uk/home-automation>
- [11] Atlas, "Make the most of your future with a smart home," 2012. [Online]. Available: <http://www.atlassmarthomes.com/>
- [12] S. o. E. E. CASAS Project and C. Science, "Daily life, summer 2009," 2009. [Online]. Available: <http://ailab.wsu.edu/casas/datasets.html>
- [13] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., July 2011, ch. Classification: Advanced Methods, p. 703.
- [14] M. Mozer, "The neural network house: An environment that adapts to its inhabitants," in *Proceedings of the American Association for Artificial Intelligence*, A. Press, Ed., 1998, pp. 110–114.
- [15] M. C. Mozer, "Lessons from an adaptive home," in *Smart Environments: Technology, Protocols and Applications*, D. J. Cook and S. K. Das, Eds. John Wiley & Sons, Inc., 2005, pp. 271–294.
- [16] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.
- [17] A.-M. Vainio, M. Valtonen, and J. Vanhala, "Proactive fuzzy control and adaptation methods for smart homes," *IEEE Intelligent Systems*, vol. 23, pp. 42–49, 2008.
- [18] J. Mendel, "Fuzzy logic systems for engineering: a tutorial," *Proceedings of the IEEE*, vol. 83, no. 3, pp. 345–377, mar 1995.
- [19] J. Lu, T. Sookoor, V. Srinivasan, G. Gao, B. Holben, J. Stankovic, E. Field, and K. Whitehouse, "The smart thermostat: using occupancy sensors to save energy in homes," in *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '10. New York, NY, USA: ACM, 2010, pp. 211–224.
- [20] A. Barbato, L. Borsani, and A. Capone, "A wireless sensor network based system for reducing home energy consumption." in *SECON*. IEEE, 2010, pp. 1–3. [Online]. Available: <http://dblp.uni-trier.de/db/conf/secon/secon2010.html#BarbatoBC10>
- [21] M. Kugler, F. Reinhart, K. Schlieper, M. Masoodian, B. Rogers, E. André, and T. Rist, "Architecture of a ubiquitous smart energy management system for residential homes," in *Proceedings of the 12th Annual Conference of the New Zealand Chapter of the ACM Special Interest Group on Computer-Human Interaction*, ser. CHINZ '11. New York, NY, USA: ACM, 2011, pp. 101–104.