# Semi-automation in Perceptive Environments: A rule-based agents proposal

Manuel García-Herranz, Pablo A. Haya, Abraham Esquivel, Germán Montoro, Xavier Alamán,

Escuela Politécnica Superior, Universidad Autónoma de Madrid,
Madrid, Spain
manuel.garciaherranz@uam.es
http://www.ii.uam.es/~mgherranz

**Abstract.** This article analyses the requirements of automation and adaptation in the so called perceptive environments. These environments are places with the ability of perceiving the context through sensors and other mechanisms. Focusing on personal/home environments, we present a first approach to semi-automatic adaptation of Perceptual Environments through a system of rule-based and configurable modular agents able to explain their behaviours and to adapt to changing habits of the users. This work has been partially funded by project TIN2004-03140 and by U.A.M-Gupo Santander (project Itech Calli).

## 1 Introduction

Since Mark Weiser first referred in 1991 to the term *Ubiquitous Computing* [Weiser, M. 1991], describing a world in which computers are seamlessly integrated into it, vanished in the background and interconnected, many problems and opportunities have arisen from that world, rich in information and interaction, and many projects have been born in the intent of bringing that world of possibilities to reality.

One of the riddles arising from this vision is how to transform these *Perceptive Environments*, able to grasp its context and to produce a coherent vision of it, into *Interactive Environments,* in which the user is able to communicate and interact with its living space, or into *Smart Places,* in which the environment itself is able to make decisions based on that perceived context. In other words the question would be how to populate the environment with *context-aware* applications and services for the benefit of its inhabitants. In fact, the benefit of the inhabitant is also a matter of context, bounded to who is the inhabitant and in what kind of environment he or she is. It is easy to see that an old woman at home and a young one at work will share little but the willing of comfort and happiness, ambiguous terms with different meanings for each of them.

Thus, having the information of context and the ability to change it what we lack is the ability for defining the willings or preferences of its inhabitants. But does the inhabitant herself have this ability? Again we can say that everybody knows what he wants or what he does not, but knowing and describing, as in the genie of the bottle tale, are not always the same thing. Hence the adaptation of the environment to the user's preferences becomes a non trivial task.

Some projects, as The Neural Network House [Mozer, M.C. et al. 1995], have decided not to ask the user but to infer directly from his behaviour what he wants or what he does not, in order to allow the house to *"program itself"* [Mozer, M.C. 1998] in the benefit of the user. For doing so, ACHE (Adaptive Control of Home Environments), as the system is called, *"monitors the environment, observes the actions taken by occupants and attempts to infer patterns in the environment that predict these actions"* using the advantages of **neural networks** [Mozer, M.C. 1998]. But as stated by P. Maes [Maes, P. 1994] there are two main problems to be solved by software agents: **competence** and **trust**. Although neural networks may achieve a high degree of competence it is hard for a user to feel comfortable in delegating tasks to a system unable to explain its behaviour.

In the artificial Intelligence Lab of the MIT a reactive behavioural system, *ReBa* [Kulkarny, A.A. 2002], has been developed for the *Intelligent Room* human-computer interaction experiment. Kulkarny has based the system on five design principles: *Context-awareness*, *conflict-resolution*, *adaptability*, *user-centricity* and *evolvability,* and he implemented it as a set of rules bounded to an *activity* in what he called an *activity bundle*. Those bundles are defined a priori by developers by identifying the activity to which a reaction is relevant. If the user wants the system to react to a new activity he must install the appropriate bundle. In order to achieve the user-centricity ReBa provides a mechanism of macros, which may be created by the user and invoked at any moment. This mechanism differentiates the "standard" reactions from the personalised ones in such a way that they will never merge, meaning that the personalised reactions will stay as patches over the standard system. Additionally, to identify the activity to which the desired reaction should be associated is not always a trivial task and it is possibly far beyond what the user is willing to do. We believe that the system should allow the user to express his desires in a simple way and it must integrate those desires with the rest of behaviours.

At Xerox Parc, one of the pioneers in Ubiquitous Computing, they define four categories of Context-aware Computing Applications [Schillit, B. 1994]: *proximate selection*, *automatic contextual reconfiguration*, *contextual information and commands*, and *context-triggered actions*. They have also implemented instances of these applications over de PARCTAB system. The most relevant for the purpose of this article are the *context-triggered actions* applications. Those actions are simple IF-THEN rules encoding a context that triggers an action. Two different applications have been implemented regarding these context-triggered actions: *Watchdog* and *Contextual Reminder*. The former allows the creation of rules over an *Active Badge* ("a tag that periodically broadcast a unique identifier for the purpose of determining the location of the wearer" [Schillit, B. 1994]) of the type:

```
Badge location event-type action
```

Where *badge, location* and *event-type* refer to the Active Badge, taking the *event-type* one of the values: *arriving*, *departing*, *settled-in*, *missing* and *attention*, and being the *action* a Unix Shell command.

Thus, an example for playing a sound message when coffee is ready would be:

```
Coffee Kitchen arriving "play -v 50 ~/sounds/ready.au"
```

On the other hand the *Contextual Reminder* allows to pop up a message according to "when, where, who and what is with you" [Schillit, B. 1994]. One of the problems of this system is the language limitation for context definition; in the *Watchdog* application we can only define the context of an Active Badge at a time. For example, there is no way to easily specify that the audio signal for the coffee maker should only be played if the TV is turned off. The same problem can be found in the *Contextual Reminder* application in which "when, where, who and what is with you" are not powerful enough to describe how things are, were or will be.

Aside from these problems, the rule-based system has proved to be an effective mechanism for expressing desires in the form of reactions to context states. Some projects have used this mechanism to other uses or they have extended its potential. This is the case of *CONON*, an OWL encoded context ontology, developed at the *Institute for InfoComm Research* and the *School of Computing of the National University of Singapore* by which they "*studied the use of logic reasoning to check the consistency of context information, and to reason over low-level, explicit context to derive high-level, implicit context*" [Wang, H.X. 2004]. For doing so CONON uses some ontology reasoning rules, defined in OWL, to describe properties such as *Transitive-Property* or *inverseOf.* Thus, if the property *location* is defined in the ontology as *Transitive¸* knowing that Wang is located in the kitchen and the kitchen is located in the first floor, the system would reason that Wang is also located in the first floor; therefore the consistency of context information is acquired automatically. On the other hand, in order to provide a mechanism for extracting high-level context from low-level information, the system allows the creation of *User-defined context reasoning rules* in such a way that he can define a rule of the type: if Wang is located in the bedroom, the bedroom light level is low and the bedroom drapes are closed then Wang is sleeping. We believe that this potential of the rules is most desirable.

One of the most critical issues in rule-based systems, in which by describing a context state it is possible to associate a reaction to it, is the ability for defining the context. The lack of this describing power will lead to bothering the user with non-wanted reactions or missing situations to react to. In this way we should mention three different projects: the *implicit Human Computer Interaction* project of the University of Karlsruhe at Germany [Schmidt, A. 2000], CONON [Tan, G.J. 2005] from Singapore and PRIMA [Brdiczka, O. 2005] from France. In [Tan, G.J. 2005], as in [Schmidt, A. 2000], an extension is presented for defining context, based in the concept of event. Distinguishing between *Primitive events* (pre-defined in the system and detectable by a mechanism embedded in it) and *Composite* ones (those being composed of primitive or other composite events) they provide a set of operators to define sequences as well as periodic and non periodic events.

In [Schmidt, A. 2000] it is defined an XML based language to describe what was called the *implicit human computer interaction*. This language uses different types of semantics for grouping contextual variables: one (match one or more of the variables of the group), all (match all of the variables of the group), none (match none of the variables of the group).Thus, when all the defined groups evaluate to true, the associated action is triggered.

These two projects extend the definition of context through sequences of events and grouping of variables. On the other hand, in PRIMA, it is defined a mean to automatically identify situations in which the described context is ambiguous and it should be perceived as, at least, two different sub-situations, one for which the associated action is right and another for which it is wrong. This way of situation splitting, provided by PRIMA, is most useful in automatically adapting predefined rules, consequently relying on the expression of desires on the user and the tuning of his definitions on the system.

Regarding the use of rules, we should mention the OCP (Open Context Platform) system, developed at the University of Murcia [Nieto, I. et al. 2006], *"a middleware which provides support for management of contextual information and merging of information from different sources"*. Based on Semantic Web technologies they developed the automatic reasoning on contextual information through a context inference mechanism based on rules of various types: do-if rules, do-for-all rules, etc. This inference is carried out using the SWRL guidelines (Semantic Web Rules Language) [Alesso, H.P. 2004] and the Jena platform as inference motor.

After this introduction is it possible to notice the increasing interest of the scientific community in automatic reasoning over contextual information and how the widespread use of rule-based systems is showing interesting results and possibilities. On the other hand no system has fulfilled all the desirable requirements for a user-centred and good-performing system as it should be the one developed for the daily living spaces of a non-technical user.

In the following section, based on the analysis of the projects presented above as well as on the user expectative, we will present the requirements that guided the development of our proposal.

## 2 Laying the foundations

Based on the popular definition of quality *"Quality is a measure for the adaptation to the use"* and following the butler, or personal assistant, paradigm we ask ourselves the following question: what does the user expect from the best-quality butler?

Following our intuition as well as others steps [Hamill, L. and Harper, R. 2006] it becomes clear that the main requirement for a butler, for now on called *agent*, is that of being **non-disturbing**, meaning that an assistant is supposed to be a help, not a workload or burden. But more than as a requirement this could be seen as an idiosyncratic principle, without which the rest will have no sense. In this way we can say that the agent should not bother the user more than she is willing to support.

On the other hand, and answering to our question, we can say that a user may want to **express** his desires to the agent in a simple and natural way. In addition he may want to ask the agent for some **explanations** about his reactions in order to understand and correct its behaviours. And last, but not least, he will expect the agent to **learn** by itself and adapt to small changes without being explicitly requested for.

Following the principles of competence and trust stated by P. Maes [Maes, P. 1994] we can say that an agent will obtain competence by acquiring explicit knowledge from the user and implicit knowledge from autonomous learning; conversely, trust will be gained through the ability to explain its behaviours in the user's language.

Since preferences may vary from user to user and through environments and situations and given that these are object to change, even more in personal environments, it is also desirable a high degree of **modularization**.

Summarizing, we can conclude that our system is based on two principles: **non-disturbing** and **modularization**, and over three requirements: **easy of expression**, **ability of explanation** and **automatic learning**.

## 3 Design principles

Given all the knowledge required for expressing the desires and preferences of the user and the conclusions obtained by the agent, it is necessary to find means for expressing it in such a way that we meet our requirements.

Since every kind of knowledge can be encoded in a language, from the inscrutable language of neurons, to music, Swedish or C++, each of them with its strengths and weakness, our task becomes more like finding the language that best describes the needs of our world of preferences, desires and conclusions and, in addition, allows us to meet our requirements of expression, explanation and learning. Although expression and learning can be met by almost any language, the explanation requirement forces us to a language capable of describing its encoded knowledge in a human readable way, thus excluding black-box alternatives such as neural networks.

On the other hand it is necessary to define what kind of knowledge is going to be encoded in that language; in this way we focus on two different issues. First and most important, we want to express the desires and preferences of the user. Since most of these desires respond to the willing of getting something done when some situation arises, we can say that we need a language capable of describing the "something to be done" and the possible "arising situations", in other words, actions and context.

Secondly, regarding the conclusions, it is also desirable to find a language with the ability for describing high-level context from low-level one. As we already noted the need for describing context, we can apply the same requirement here.

Finally we conclude the need for a language capable of describing various contexts and actions, able to associate them and, in addition, being human-readable. These two main requirements have leaded us to the choice of a **rule-based** system.

Alternatively, one of the main problems of context-aware applications responds to when to supervise the system, that is, when to check the context in order to act as expected. Because supervised environments can grow to considerable sizes and given that not every component has the same timing constraints, determining the time intervals for checking the state of the context becomes a tricky problem. Thus we have decided to take an **event-based** approach in which we assume that only a change in the environment can trigger another change on it. We believe that this is true for everything but time itself –i.e. I may want to turn on the washing machine at

22:00 – for what we will need a way to model the time into context or treat it in a different way.

## 4 A language for Perceptive Environments

Following the design principles stated in the previous section we have developed a language of rules for a modular agent system. In this manner the system will be composed of independent agents each of which comprises a set of rules encoding a set of reactions to context states.

Each rule is composed by three parts: *trigger*, *conditions* and *action*, and has an associated weight. This weight is used as a measure of trust, modified by a learning process of reinforcement. When the weight of an active rule falls over a certain threshold a deactivation process takes place i.e. deactivating the rule or notifying the user depending on the disturbing preferences of the user. Contrary, when a rule becomes deactivated it continues its execution without applying its action, thus continuing with the reinforcement process. If, in contrast, the weight of a deactivated rule surpasses a certain threshold, an activation process, similar to the deactivation one, is applied.

Regarding the internal structure of the rules it is possible to distinguish between three different parts:

*Trigger*: a supervised context variable responsible of activating the rule.
*Conditions:* a set of pairs "context variable-value" representing a context state that needs to be satisfied for detonating the action.
*Action:* a pair "context variable-value" to be set when in a triggered action all its conditions evaluate to true.

Most of the analysed systems only differentiate between conditions and actions, using the former as triggers. We have observed that in many situations not every condition to be met should trigger the action. As an example let's imagine a living room with a TV and a dimmer light with three values: HIGH, LOW, OFF; in this scenario, when turning on the TV, we want to set the light to its LOW value if it was in the HIGH one, leaving it LOW or OFF if in any of them. This behaviour can be encoded into the context TV-ON AND LIGHT-HIGH and the associated action LIGHT-LOW. Given that context, when the light status is HIGH and the user turns ON the TV then all the conditions evaluate to true and the action is triggered, that is, the light is set to LOW level. But since no triggers are specified, if the TV was already turned ON and the user, looking for more light, set the light level to HIGH then all the conditions will evaluate again to true and the system will change the light level to LOW, contrary to the user's desires. This example shows the necessity for differentiating between conditions and triggers.

Although the user may want to specify more than one trigger or action per behaviour we have chosen a system in which the rules only support one trigger and action. Every behaviour with more than one trigger or action can be encoded using several rules each of which will have only one trigger and action. This mechanism simplifies by far the task of identifying wrong rules in the learning process.

All these behaviours, encoded into rules, are stored into modular agents. Thus it is possible to create, enable or disable different sets of behaviours corresponding to different users, periods, situations or devices without affecting the rest of the system. Since the Blackboard system of Interact [Haya, P.A. 2004], over which the work described in this paper is implemented, already deals with collisions of actions from different agents over an entity, the agent will only have to manage collisions of its own rules. This is much simpler to do and, since most of the times an agent is associated to a user, it can easily deal with user's preferences.

## 5 Prototype

Based on the architecture presented above we have implemented a first prototype of the agent system that, as a first approach, deals with the three requirements stated in section 2: **easy of expression**, **ability of explanation** and **automatic learning**.

The former is firstly achieved through the almost-natural language of rules described in section 4. This language uses some especial characters to differentiate between trigger, conditions and action (see Code Section 1). The trigger can be a change on a property or on a relation of an entity; the conditions are of the form *element operator element* where an element can be an *entity*, a *property* of an entity, a *relation* or a mere *value* and the operator can be *equal*, *not-equal*, *greater*, *smaller*, *exists* or *non-exists*. Of course not every operator can be applied to every type of element and the agent will throw exceptions for bad-formed conditions. Finally, the actions are composed by an *element,* an *operation* and another *element.* The operation could be *assign*, *assign-opposite-value*, *minus*, *add*, *create* or *delete.* As with the conditions the agent will check for the good formation of the actions.

```
tv:tv1:status :: tv:tv1:status = ON ; dimlight:lampv1:value > 20
-> dimlight:lampv1:value := 20
```

Code Section 1: example of a rule following the norm *Trigger :: condition ; condition ; ... -> action.* The trigger is the status of the TV, meaning that the rule will be triggered whenever this status changes. This rule has two conditions: 1) the TV is turned on and 2) the dimmable lamp is set to a "high" value. The action is to set the lamp value to "medium". In result, when the TV status changes and the new status is "ON" -meaning "when turning on the TV"-, if the dimmable lamp is set to "high" then lower it to "medium".

In this first approach the explanation requirement is accomplished through a mechanism of on-request traces by which the system shows the user its internal process of learning, triggering, evaluating and reacting as the relation between the context changes and its rules inference process (see Code Section 2)

```
- Changed property: tv:tv1:status to value: OFF
```

```
Triggered rules:
   - [ ] Rule 20: tv:tv1:status :: tv:tv1:status = ON (false)->
   light:lamp_1:status := OFF (10)
   - [ ] Rule 21: tv:tv1:status :: tv:tv1:status = OFF (true)->
   light:lamp_1:status := ON (10)
   - [ ] Rule 22: tv:tv1:status :: tv:tv1:status = ON (false); dim-
   light:lampv1:value > 20 (false) -> dimlight:lampv1:value := 20 (10)
   - [ ] Rule 23: tv:tv1:status :: tv:tv1:status = OFF (true); dim-
   light:lampv1:value = 20 (false) -> dimlight:lampv1:value := 35 (10)
 Applied rules
   - [ ] Rule 21: tv:tv1:status :: tv:tv1:status = OFF ->
   light:lamp_1:status := ON (10)
      Property light:lamp_1:status set to ON
```

Code Section 2: example of the explanation mechanism of traces for the lamp_1 turning on when shutting down the TV. The header corresponds to the change responsible of the process (the property *status* of the TV changing to OFF). Then it appears a list with all the rules triggered by this event (all the rules having the property *status* of the TV within its triggers). Finally the list with the applied rules appears in the last place. These rules are those whose conditions evaluate all to true. In this case the rule 21, whose only condition "the TV is OFF" cause the lamp *lamp_1* to be turned on. At the end they appear all the changes made by the rules. Additionally every rule begins with a checkbox determining if it is activated ([ ]) or deactivated ([X]) and its rule number -i.e. Rule 21.

Regarding the learning process, every rule has a weight that reflects its confidence. Every time a rule is triggered and its action executed the system marks it for feedback training; if within a fixed length of time (currently one minute) no other process contradicts the action executed by the rule then it is rewarded positively by incrementing its weight, otherwise the value is decremented.

Finally, all these processes are chained into the following algorithm:

```
For every change in the context
  If is due to an agent order then store the rule index
  Educate all rules except the indexed one
  Update the true-false value of the conditions
  Obtain the triggered rules
     If their conditions evaluate all to true
Send its actions to the Blackboard
```

Code Section 3: Algorithm for execution of rules according to a context change

As it can be seen in Code Section 3 the value of the conditions is updated every time the context changes. Another alternative would be to check the value of the conditions only when the rule is triggered, thus alleviating the communication of the system. Even though, we have decided the former so the computational load gets spread all over the time and, when triggering a rule, the information is already available, reducing the decision time. This response time is most of the times a critical issue i.e. a delay of one second in turning on the lights will ruin the success of the application. Additionally, the extra communication of this alternative grows to cero as the number of rules is increased.

This prototype has been implemented over a real environment (a living room and an office space, see Figure 1) and has allowed replacing multiple ad-hoc applications such as the switch-light control or access control and developing new applications such as enhanced switch and automatic light control in a simple, comprehensible and fast manner with very satisfactory results. By the time of this article these applications are distributed within two different agents, one in charge of extracting the localization of the inhabitants from various low-level context variables and the other, a butler, responsible of light, access and comfort control as, for example, lowering the lights when watching TV, saluting when entering, informing when coffee is ready or directing the inhabitants pictures to the appropriate displays.

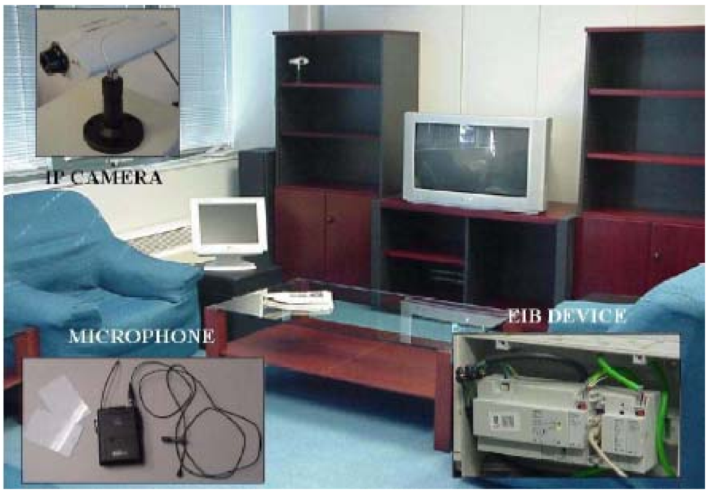

Figure 1: Snapshot of the living room and details of some system parts and components: EIB domotic control bus, IP camera and wearable microphone for input data, dynamic photo-frame and multimedia network of speakers and TV set.

## 6 Conclusions

From this first approach we should remark some points of special interest. Firstly we should point out the wide possibilities of description and easy of programming brought by the use of triggers in the rule language. This simple and intuitive concept has been a keystone in the development of the system. Secondly we have found quite interesting the modularization of agents, bringing the possibility to easily adapt the environment to new inhabitants and situations by enabling or disabling agents.

In relation with the challenge of learning automation we have found that a semi-automatic approach in which the learning is based on knowledge given by the user is much simpler and effective and less annoying for him. Future work is being focused on the use of weight oscillations to identify errors, thus if a weight is increased and decreased systematically it could be as a consequence of a too general set of conditions. We believe that the path of automation in Perceptive Environments should go through enhancing human control, not over passing it, consequently future systems will require means for naturalizing and extending their control mechanisms, helping the user to know and express his desires. In this sense, learning should play a tuning and suggesting role in the system, an aid to deal with the genie of the bottle.

## References

ALESSO, H.P. and SMITH, C.F. 2004. Developing Semantic Web Services. A K Peters, Ltd. ISBN 1568812124.

BRDICZKA, O.; REIGNIER, P. and CROWLEY, L.J. 2005. Supervised Learning of an Abstract Context Model for an Intelligent Environment, Smart Objects and Ambient Intelligence. SOC-EUSAI 2005 (Grenoble 2005)

HAMILL, L. and HARPER, R. 2006. Talking Intelligence A Historical and Conceptual Exploration of Speech-based Human-machine Interaction in Smart Homes. International Symposium on Intelligent Environments (Microsoft Research, Cambridge, United Kingdom, Apr 5-7, 2006), 121-127

HAYA, P.A.; MONTORO, G. and ALAMÁN, X. 2004. A prototype of a context-based architecture for intelligent home environments. International Conference on Cooperative Information Systems (CoopIS 2004), Larnaca, Cyprus. October 25-29, 2004. 477-491.

KULKARNY, A.A. 2002. A reactive behavioral system for the intelligent room. Massachusetts Institute of Technology.

MAES, P.; 1994. Agents that Reduce Work and Information Overload. Communications of the ACM, 7, 37, 31-40

MOZER, M.C.; DODIER, R.H.; ANDERSON M.; VIDMAR L.;, CRUICKSHANK III, R.F. and MILLER D. 1995. The Neural Network House: An overview. Current trends in connectionism, L. Niklasson and M. Boden, Eds., Lawrence Erlbaum, Hillsdale, NJ, 371-380.

MOZER, M.C. 1998. The neural network House: An environment that adapts to its inhabitants. In Proceedings of the AAAI Spring Symposium on Intelligent Environments (AAAI98).

NIETO, I.; BOTÍA, J.A. and GÓMEZ-SKARMETA, A.F. 2006. Information and hybrid architecture model of the OCP contextual information management system. Journal of Universal Computer Science, 12, 3, 357-366

SCHMIDT, A. 2000. Implicit Human Computer Interaction Through Context. Personal and Ubiquitous Computing. 2/3, 4

SCHILIT, B.; ADAMS, N. and WANT, R. 1994. Context-Aware Computing Applications. Workshop on Mobile Computing Systems and Applications (IEEE, Santa Cruz, CA, US)

TAN, G.J.; ZHANG, D.; WANG, X. and CHENG, S.H. 2005. Enhancing Semantic Spaces with Event-Driven Context Interpretation. In Proceedings of Pervasive Computing, Third International Conference. (PERVASIVE, Munich, Germany, May 8-13, 2005), 80-97

WANG H.X.; ZHANG, Q.D.; GU, T. and PUNG, P.H. 2004. Ontology Based Context Modeling and Reasoning using OWL. In Proceedings of PerCom 2004 (Orlando, FL, USA, March), 18-22

WEISER, M. 1991. The computer for the 21st century. Scientific American, 265, 3, 94-104