Easing the Smart Home: Translating human hierarchies to Intelligent Environments

Manuel García–Herranz, Pablo A. Haya, and Xavier Alamán

AmILab, Ambient Intelligence Laboratory, E.P.S Universidad Autónoma of Madrid
C/ Francisco Tomás y Valiente 11,28049 Madrid, Spain
{manuel.garciaherranz,pablo.haya,xavier.alaman}@uam.es
http://amilab.ii.uam.es

Abstract. Ubiquitous computing research have extended traditional environments in the so–called Intelligent Environments. All of them use their capabilities for pursuing their inhabitants's satisfaction, but the ways of getting it are most of the times unclear and frequently unshared among different users. This last problem becomes patent in shared environments in which users with different preferences live together. This article presents a solution translating human hierarchies to the Ubicomp domain, in a continuing effort for leveraging the control capabilities of the inhabitants in their on–growing capable environments. This mechanism, as a natural ubicomp extension of the coordination mechanism used daily by humans, has been implemented over a real environment: a living room equipped with ambient intelligence capabilities, and installed in two more: an intelligent classroom and an intelligent secure room.

Key words: Ubiquitous Computing, Human-centered computing, Rule-based processing, Command and control

1 Motivation

Once environments have been supplied with perceiving and actuating capabilities, lots of research efforts have been put into creating "intelligent environments" to actuate on the user's behalf. In this sense, two main trends of research can be identified: *autonomous environments* and *automatic environments*. The former represent those systems that, without intervention of the user, try to reach the goals they are intended for (such as reducing energy consumption [1] or minimizing the number of tasks the user has to do [2]). Many of these systems are based on "black box technologies", meaning that they are not human–readable, such as neural networks, Hidden Markov models or Bayesian networks. Even though these last ones may have an interpretable graphical representation being, they are not strictly readable. Automatic systems, on the other hand, represent those systems trying to replicate the solutions that have been explicitly given by the user. These systems are normally based on "white box technologies" such as rule–based [3] or case–based [4] expert systems, since, as stated by Myers, *the closer the language is to the programmer's original plan, the easier the refinement process will be* [5]. Closer or farther from the end–user mental plans, those

systems pretend to be programmable and, to that extent, they provide means for creating, structuring and organizing "code". In this sense, Myers points at rule-based languages as the ones naturally used by users in solving problems [5]. Summarizing, while autonomous environments try to "replace" the user, automatic environments try to extend their control.

While both approaches have advantages and disadvantages, they share the problem of conflicting inputs due to more than one inhabitant i.e. collisions. While autonomous solutions have a hard time to learn from a "noisy" environment, automatic approaches need some clarifying input for what to do in conflict situations.

Our choice has been to leverage the user's control over the environment through an automatic approach, focusing on personal environments and trying to **get as close as possible to the end–user**. While a more detailed description of our overall choice can be found in [6], this paper focuses on its capabilities for dealing with social conflicts.

2 Indirect control in intelligent environments

The purpose of the indirect control mechanism is to allow users to "program" behaviors/preferences/implicit interaction on their environments. For doing so, we use a rule–based agent mechanism that will be briefly explained to provide the basis for the rest of the article.

2.1 A rule–based multi–agent programming system

As in every intelligent environment, the first step is always that of perception (see Figure 1(a)), done in our approach through a middleware layer ("the Blackboard") in which every element of the environment, either real or virtual, is represented as an *entity* with *properties* and *relations* between them [7]. Every change on the environment is reflected in the blackboard and vice versa.

Over this context layer there is an interaction layer in which all the applications are located, the indirect control mechanism among them [6]. This mechanism is designed to allow users to program their environment and is based on a rule–based core language. Rules can be created and organized into agents (the minimal self–sufficient structure). Agents are represented in the blackboard as agent entities with a *status* and *task* properties and the *is_owner*, *located_at* and *affects* relations, linking it with the user that create it, the location it acts in and the elements it affects, respectively. Following Paperts' ideal of "low–threshold no ceiling" [8] the rule–based language is based in a basic language with some expression extensions. The basic language has three parts: *triggers*, *conditions* and *actions*, in analogy with the natural *"When ... if ... then ...* structure.

3 Human hierarchies

Through this system, the diversity of preferences and tasks coexisting in the environment, is reflected in a multitude of context–aware applications running

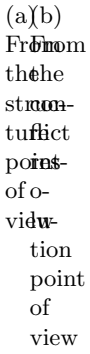

Fig. 1. AmiLab layers. The World layer (i.e. what is in the environment), the Context layer (i.e. how ubicomp acces it: abstraction [7] and privileges [9]) and the Interaction layer (i.e. how can be used: Explicitly and Implicitly). This article focus on Implicit interaction programmed by the user.

over the same environment. In traditional environments, these preferences and tasks are prioritized according to some hierarchies accepted by the environment inhabitants. Analogously, since context–aware applications are automated user preferences, they must have a mechanism through which to represent and apply these same hierarchies if they are to coexist in the same environment.

Hierarchies are the natural social structures to establishing an order of preference but, linked to the social group that created them, they are multiple an dynamic and their complexity reflects the complexity of the social group they rule. In addition, every social group has its own ways for generating and accepting their hierarchies.

This work focus on social groups as the only generators/acceptors of their hierarchies, looking forward to enrich the control users' experience over their environments, rather than proposing a fixed social engineering solution.

In this sense, our indirect control mechanism is just an extension of the user's control. Decisions are automatic in the sense that they do not need direct human intervention but they were created by the end–user and should be governed by the same hierarchies governing users' daily life.

Given that every action in the environment is –at least indirectly– produced by a human and our believe that each social group generates/accepts its own hierarchies, this work is based on the principle that **neither the system automatically nor a third human party such as an engineer should specify the hierarchies** governing a social group, but they should allow instead particular social groups to express their own. This apparently naive principle presents a profound problem when designing a solution since any fixed structure will interfere with the possibly different natural structure of the user and, secondly, the flexibility of the system must not interfere with the simplicity of end–user oriented solutions. Both structure and flexibility must adapt to the ways humans

use to create and organize their hierarchies. In this direction we will analyze two social factors: **purpose** and **complexity**

3.1 Purpose and complexity

As each type of conflict produces a different hierarchy domain, we find that the *purpose* for what hierarchies are build is varied and entangled. Social hierarchies (i.e. *who goes first?*) and task hierarchies (i.e. *What goes first?*) are just two examples that can be in conflict: What happens if there is a conflict between a high–priority person doing a low–priority task and a low–priority person doing a high–priority task? A new hierarchy (or an exception) will be born: A hierarchy of hierarchies, so to speak. Some systems have attacked this problem through classifying, organizing and structuring. Kakas et al. [10], for example, provide each agent of a multi–agent system with two types of priority rules: *Role priorities* and *Context priorities*. Role rules prioritize according to the *role* of the parts involved (some sort of social hierarchy), while context rules prioritize role rules according to some extra context (some sort of task hierarchy). Additionally, each agent is supplied with a *motivation* structure, a hierarchy on the goals it pursues. This goals are defined according to Maslow's need's categorization [11]. Finally,an additional hierarchy is specified to define the agent's "personality" (i.e. his decision policy on needs to accomplish the goals of its motivation). While this kind of structuring captures many of the flavors of human behavior, it presents some problems when applied to personal environments. First, it needs a professional (familiar with Maslow's theory, to begin) to program the system, and a clear a priori classification of the family goals, roles and tasks. This engineering solutions, while perfectly valid for domains such as business automation, should not be applied to home environments in which a programming third party, a too rigid categorization or the need of a deep a priori definition breaks some of Davidoff's principles for smart home control, such as "allow an organic evolution", "easily construct/modify behaviors", "Understand improvisation" or "participate in the construction of family identity"[12]. Thus, hierarchies should no be fixed to specific domains (e.g. roles or goals) neither presume the knowledge of complex concepts (e.g. Maslow theory).

Secondly, as a social structure, we considered *complexity* the second factor of interest while analyzing hierarchies. How do we allow the different degrees of complexity of different social groups? How can we measure those different degrees of complexity? Y. Bar–Yam [13] *interdependence* and *scale* concepts to measure complexity are quite useful in this. Interdependence describes the effects a part has over the rest of the system: "If we take one part of the system away, how will this part be affected, and how will the others be affected?". Scale, on the other hand, refers to the different degrees of complexity a system acquires depending on how close or far removed apart is the observer. These concepts characterize complex systems: both a microprocessor (a designed system) and the global economy (an spontaneous system) have millions of components but, while the former is easier to understand (the interdependence is clear), the latter is not fully understood nor controlled by anybody (unclear interdependence). On

the other hand, the former's growth depends on a new design while the latter is robust and adaptive to changes in its subcomponents, to name some of the consequences of their complexity's idiosyncrasy.

Social networks range from strongly planned, as in military hierarchies, to highly spontaneous, as in the Internet, depending on the goal of the structure. Some have their complexity spread among the structure (e.g. the Ford T production chain) while others condense it in specific parts (e.g. a diamond cutting business). The different degrees of interdependence and scale a solution provides is easy to see and clearly shows what kind of complexities can be achieved with it.

Finally, from the necessity of creating flexible structures and hierarchies, a decentralized mechanism for hierarchy management is deduced. A mechanism of this kind, in which there is not necessarily a central agent coordinating the processes does not prevent the creation of centralized hierarchies but does not impose them either. As an homomorphism with natural hierarchies, individuals can create their own hierarchies and, if they want, coordinate them with those of other individuals or centralize them, according to the problem they are designed to solve.

3.2 Translation to intelligent environments

Any coordinating structure, such as hierarchies, is strictly bounded to the nature of the blocks it coordinates. Thus, we should refresh some of the design principles of the indirect control mechanism, to understand how hierarchies are extended and applied to it:

- Decision rules are grouped in sets: the agents explained in section 2.
- Agents are represented in the blackboard layer [7] and can be activated/deactivated through it.
- In order to replicate the natural organization of preferences, agents must *belong to* a user or group of users and may be tagged with the *activity/purpose* they are designed for and *located* in the place they affect.
- To keep track of the natural responsibility chain, agents must be related with the elements *affected* by their rules.

3.3 Hierarchies structure and definition: Multilayer filter

Once decided not to impose any kind of structure to the users when creating their hierarchies, and once defined the indirect control blocks' structure, we must define the means by which users are to express their hierarchies. In a rough approach we distinguish two kinds of conflicts in the ubicomp domain: those with users directly involved and those without them. While in the former technologies can help them to ease their policies, in the latter technologies are the only mean for users to apply any policy. Thus, we will present our solution paying especial attention to hierarchy automation in the indirect control.

According to the nature of conflicts, we have implemented a layer structure to solve conflicts (See Figure 1(b)) acting as a series of filters between the user desire and its effect in the world. In doing so, the AmILab layer structure (see Figure 1(a)) has been used to deal with conflicts of different nature, mainly **ownership** and **collisions**. Collisions can be classified in context–dependent and context–independent, according to whether the only important factors to solve the collision are the elements colliding and the point of collision or whether there is any other relevant element in the resolution. This last type of conflict is the one with more elements involved in the solution. The other two can be solved in the *Context layer*: ownership conflicts in the *Privacy layer* and context–independent collisions in the *Blackboard*, respectively. Context–dependent collisions (the focus of this paper) must be solved in the interaction layer for what we propose the use of the same rule–based agents mechanism mentioned in section 2. Allowing end–users to **use the same programming structure they use to program their preferences to program the hierarchies to control that preferences**.

Even though any hierarchy can be programmed through the rule–based agent mechanism, ownership conflicts and context–independent collisions can be solved in the context layer too, being more natural for their purpose. The choice will depend on the users and, among other things, will reflect the degree of interdependence and scale of their natural hierarchy (see section 3.1).

Firstly, the Blackboard layer provides a priority queue to each element [14] with a default policy to apply in case of collision (i.e. if despite the rest of the layers a collision reaches this point, the default policy is applied).

Secondly, the Privacy layer allows users to establish access rights to the elements they own [9]. Besides being used as a privacy filter it helps in constructing hierarchies where the owner is the only relevant factor in the policy (i.e. users control freely the access to their elements).

Finally, through the Interaction layer, users can create structures to control the environment that are, in turn, represented as part of the environment. In this way they can create agents whose rules control the status of another agent (instead of a physical element of the environment). Similarly, a rule can be used to set the privacy preferences. This mechanism to control indirect control structures allows the creation of hierarchies in many levels as desired. The complexity of the system –interdependence and scale– is up to the inhabitants and their natural hierarchies. Interdependence is easy to see (while not always to understand) in the Blackboard as the graph created by all the relations *affects*. Depending on the scenario this graph will range from pyramidal structures to unconnected graphs or entangled networks. With a person/s behind each agent, an element of the environment in each leave of the graph and, in between, a complex structure of conditions that, as a whole, governs the overall automatic behavior of the environment. Scale, on the other hand, can be appreciated in the different levels in which hierarchies can be expressed. To illustrate it with an example, lets consider two users sharing a house. User A prefers the light level to be low while user B prefers it high. In this situation, they can control their preferences through a single agent (associated to both of them) in which three

rules codify their preference: "if user A is in the house but not B then set the light level to low", "if user B is in the house but not A then set the light level to high" and if "both A and B are in the house set the light level to medium". Conversely, they can have an agent for each of them codifying their personal preferences, another shared agent codifying their mutual preferences (i.e. what they want when they are together) and a meta–agent deactivating their personal agents and activating the shared one when both of them are in the house and vice versa. Or, finally, they can do without agents and establish a default policy in the Blackboard (since no other factor but themselves and the light is present in the conflict) to establish *the average* as the desired value for the light when a conflict arise. While codifying the same behavior, the three approaches present a different scale and they will be preferred over the others according to the idiosyncrasy of the social group. Thus, the former will be more frequent in situations in which most of the preferences are shared (e.g. a marriage sharing a house), the second when each individual normally decides alone and some coordinating mechanism is required (e.g. student roommates) while the third one is more natural to sporadic environments in which personal preferences are secondary (e.g. a laboratory hallway). These structures have been observed in the three real environments in which the system is deployed: a simulated living–room in the AmILab laboratory (Autonomous University of Madrid, Spain), a simulated security chamber in Indra's facilities (Madrid) and an intelligent classroom in the Itechcalli laboratory (Zacatecas, Mexico).

4 Conclusions

To face the problem of having diverse inhabitants with different preferences, automatic environments need to provide means through which to **create coordination structures**. Personal environments present an specific domain in which some extra principles, such as the construction of family identity or an organic evolution, have to be consider.

To deal with the diversity of environments and social groups, the coordination mechanism has to be **flexible** enough to adapt to different social structures. We pointed at interdependence and scale (i.e. the complexity measures of Y. Bar–Yam [13]) as useful measures to characterize the adaptability of a system to different social organizations.

Finally, regarding the end–user programming factors, we stated that **the user must be able to create her own hierarchies**, excluding any third human parties or automatic systems from this process. This forced us to get as close as possible to the end user, for what three factors have been of main importance: a flexible and **multiple categorization of agents** allowing concepts such as ownership, location, task, goal or effect without imposing any; a **multi–layer structure** to deal with specific types of conflicts easily, but not restricting conflict's solutions to a specific layer and, finally, the **use of the same programming structures** with which users program their preferences to program the hierarchical structures.

Acknowledgments. Thanks to Gemma de Castro and Dominik Schmidt for their contribution to this work. This work was partially funded by the Spanish Ministry of Science and Technology through the HADA project(TIN2007-64718) and by the chair UAM–Indra of Ambient Intelligence.

References

1. Mozer, M.M.: The neural network house: An environment that adapts to its inhabitants. In: Procs of the AAAI Spring Symposium on Intelligent Environments, AAAI Press (1998)
2. Cook, D.J., Youngblood, M., Heierman, E., Gopalratnam, K., Rao, S., Litvin, A., Khawaja, F.: Mavhome: An agent-based smart home. In: Procs. of the IEEE International Conference on Pervasive Computing and Communications. (2003) 521–524
3. Bischoff, U., Kortuem, G.: Rulecaster: A macroprogramming system for sensor networks. In: OOPSLA Workshop. (2006)
4. Brdiczka, O., Reignier, P., Crowley, J.L.: Supervised learning of an abstract context model for an intelligent environment, smart objects and ambient intelligence. In: SOC-EUSAI 2005, Grenoble 2005. (2005)
5. Myers, B.A., Pane, J.F., Ko, A.: Natural programming languages and environments. Commun. ACM **47**(9) (2004) 47–52
6. García-Herranz, M., Haya, P., Esquivel, A., Montoro, G., Alamán, X.: Easing the smart home: Semi-automatic adaptation in perceptive environments. Journal of Universal Computer Science **14**(9) (2008) 1529–1544
7. Haya, P.A., Montoro, G., Alamán, X.: A prototype of a context-based architecture for intelligent home environments. In: International Conference on Cooperative Information Systems (CoopIS 2004). Volume 3290 of Lecture Notes in Computer Science (LNCS)., Larnaca, Cyprus (October 25-29 2004)
8. Papert, S.: Mindstorms: Children, Computers, and Powerful Ideas. Basic Books, New York (1980)
9. Esquivel, A., Haya, P.A., García-Herranz, M., Alamán, X.: Managing pervasive environment privacy using the "fair trade" metaphor. In: International Workshop on Pervasive Systems, PerSys 2007. (2007)
10. Kakas, A.C., Moraitis, P.: Argumentation based decision making for autonomous agents. In: AAMAS, ACM (2003) 883–890
11. Maslow, A.H.: Motivation and Personality. Harper, New York (1954)
12. Davidoff, S., Lee, M.K., Yiu, C., Zimmerman, J., Dey, A.K.: Principles of smart home control. In Dourish, P., Friday, A., eds.: Ubicomp. Volume 4206 of Lecture Notes in Computer Science., Springer (2006) 19–34
13. Bar-Yam, Y.: Analyzing the effectiveness of social organizations using a quantitative scientific understanding of complexity and scale. NECSI (May 2007)
14. Haya, P.A., Montoro, G., Esquivel, A., García-Herranz, M., Alamán, X.: A mechanism for solving conflicts in ambient intelligent environments. Journal Of Universal Computer Science **12**(3) (2006) 284–296