Evaluating social choice techniques into intelligent environments by agent based social simulation

Emilio Serrano , Pablo Moncada, Mercedes Garijo, Carlos A. Iglesias

A B S T R A C T

The primary hypothesis stated by this paper is that the use of *social choice theory* in *Ambient Intelligence* systems can improve significantly users' satisfaction when accessing shared resources. A research methodology based on *agent based social simulations* is employed to support this hypothesis and to evaluate these benefits. The result is a sixfold contribution summarized as follows. Firstly, several considerable differences between this application case and the most prominent social choice application, political elections, have been found and described. Secondly, given these differences, a number of metrics to evaluate different voting systems in this scope have been proposed and formalized. Thirdly, given the presented application and the metrics proposed, the performance of a number of well known electoral systems is compared. Fourthly, as a result of the performance study, a novel voting algorithm capable of obtaining the best balance between the metrics reviewed is introduced. Fifthly, to improve the social welfare in the experiments, the voting methods are combined with cluster analysis techniques. Finally, the article is complemented by a free and open-source tool, *VoteSim*, which ensures not only the reproducibility of the experimental results presented, but also allows the interested reader to adapt the case study presented to different environments.

1. Introduction

Ambient Intelligence (AmI) focuses on adapting to people's needs and particular situations. This is possible by incorporating omnipresent computing elements that communicate ubiquitously among themselves [61]. The interaction between human beings and these computing elements allows AmI systems to gather context-information in a dynamic and distributed way. This information is used to offer services that support daily user tasks in a smooth and adaptive way. Therefore, AmI systems need to be aware of users' preferences, intentions, and needs [19] to offer different services whose main goal is to augment their quality of life.

Some examples of applications of the AmI paradigm introduced above are: designing office spaces that smoothly move information between displays, walls, and tables; and, learning to customize lighting and temperature based on learned resident preferences [19]. These services raise an important question: what happen when resources are shared and there are conflicts between users' preferences? In the scope of the research projects THOFU [5] (Technologies for the future of hotels) and CALISTA [1] (Agent Technologies and Engineering Services Diagnostics and Configuration of Home Network using

a Mobile Phone), the authors have observed that there are plenty of scenarios where AmI systems not only have to offer a good service considering users' preferences, but also to make a decision in an attempt to maximize the users' welfare when they access shared services. For example, an intelligent hotel may offer a number of configurable and shareable services: screening rooms, decoration based on dynamic decorative panels, dance clubs, heated swimming pools, etcetera. In these cases, AmI systems have to decide how to configure these services (in the given examples: the projected film, the decoration, the music and the pool temperature). And this decision involves trying to make hotel customers as happy as possible while avoiding "resource starvation" (which, in this context, means that some customers' preferences are constantly denied in favor of the common good). As a result, AmI services have to reach consensus trying to maximize users' satisfaction.

Although this issue of accessing shared resources has not been explored in the AmI specialized literature, fortunately, *agreement technologies* (ATs) [34] have studied it in depth. ATs deal with technologies for practical application of knowledge in order to reach agreements automatically. ATs have covered a large variety of negotiation aspects such as: multi-issue negotiations, concurrent negotiations, strategy-proof mechanisms, argumentation, auctions, voting, etcetera [33]. In this scope, the use of *social choice* theory, which is concerned with the evaluation of alternative methods of collective decision-making [10], appears as a straightforward solution because its primary goal is to make a group decision.

However, the proposed solution to the problem addressed in this paper an introduced above, the social choice theory, has mainly focused on theoretical works which deal with political elections [47]. Therefore, there are a number of dilemmas to be solved in this scope: what are the benefits of using a voting system in an intelligent environment?; what are the most suitable voting systems?; and, what differences does this case present when compared to political elections? Since "there is no one silver bullet", AmI developers have to choose several of these agreement technologies to be evaluated in the specific AmI system. In this paper, the use of *Agent Based Social Simulation* (ABSS) is proposed to solve the aforementioned dilemmas.

ABSS, proposed to solve a number of dilemmas in the novel application of social choice theory for AmI systems, has become one of the most popular technologies to model and study complex adaptive systems. For example, ABSS has already been employed to study users in AmI systems [28,50]. ABSS combines computer simulation and social science by using a simple version of the agent metaphor to specify single components and interactions among them. This paper follows Gilbert and Troitzsch's [30] methodology to study systems by ABSS. This methodology involves: (1) studying the target system, (2) modeling it, (3) implementing a simulation, and (4) studying the results after executing the simulation. This methodology is interesting in this context for two main reasons. Firstly, it pays the necessary attention to the model building, i.e. the implementation of the model. Other widely used methodologies in social simulation omit this step, such as the one proposed by Fishwick [24,25]. Secondly, it is the most popular methodology for developing social simulations [21]: Gilbert and Troitzsch's book [30] has been cited over 1900 times and it is the obligatory reference in the ABSS field.

The main contribution of this paper is the use of ABSS to support the hypothesis that the social choice theory application in AmI systems can improve considerably users' satisfaction when accessing shared resources. This involves studying the differences between this application case and the most prominent social choice application, political elections. One of the main and more significant differences found is that, since there can be several services of the same type in the environment, the voter can choose the group to vote in. This allows the AmI system to use *machine learning techniques* [31] to suggest the best group in order to maximize the global satisfaction. Besides, different metrics are proposed to evaluate a number of voting systems in the AmI case such as: accumulated satisfaction, number of users with wanted configurations, or the longest wait to get a wanted configuration. These metrics are employed to study the suitability of well known voting methods such as: plurality voting, Borda voting, range voting, approval voting, and cumulative voting. Additionally, a novel voting algorithm called *exchange of weights voting* is presented to maximize the metrics reviewed. The experimental results are given in the scope of an intelligent hotel where users can share TV screens in the hall. Finally, a free and open-source tool called *VoteSim* is given to guarantee the reproducibility of these results and to allow developers to evaluate social choice techniques for different environments.

Note that the central contribution is not defining new social choice techniques but considering them in the scope of AmI systems which, among others, requires defining a novel generic agreement service for shared resources (see Section 3) and modeling it with ABSS techniques (see Section 4). Note also that although the only innovative voting system introduced in this paper is the *exchange of weights voting* (explained in Section 6.1), the remaining voting systems (defined in Section 4.2.1) have been reconsidered under a common agreement design pattern, formalized using this pattern, and implemented in a free and open-source tool.

The paper is organized following the Gilbert and Troitzsch's methodology introduced above. After revising the background in Sections 2, 3 studies a general agreement service for AmI environments, Section 4 proposes a model to simulate this target, Section 5 deals with its implementation by VoteSim, and Section 6 presents the experiments conducted with the simulation and the results obtained. The paper finishes with the conclusions and future works in Section 7.

2. Background

This paper proposes the use of agreement technologies to cover an open problem in AmI and, more specifically, the use of social choice. Additionally, the use of simulations is proposed as a method to decide the specific social choice algorithm to be used and how to optimize the global satisfaction before deploying the final systems. Therefore, this background consists of three main blocks: (1) an introduction to agreement patterns; (2) related works in social choice; and, (3) simulation platforms conceived to evaluate applications in AmI environments. It is important to note that, since both the agreement technologies and AmI systems are usually implemented by *multi-agent systems*, most related works fall into this category.

2.1. Agreement technologies

In the last years, negotiation has been covered by plenty of literature from various fields: anthropology, psychology, sociology, law, economics, business management, etcetera. Without a common framework to discuss negotiation based solutions, engineers work becomes a laborious process of trial and error. To solve this problem, agreement technologies arise as a recent discipline that can be defined as: technologies for practical application of knowledge in order to reach agreements automatically. That is, it emphasizes the formalization of knowledge, structures, protocols, and algorithms for agreements in open and dynamic environments without dictating the underlying technologies [32].

With this spirit, Iglesias et al. [32] have defined a set of agreement patterns inspired by the well-known software design patterns [49]. These authors advocate the reuse of software engineering technologies principles with proven effectiveness for modeling agreements. They propose a template for defining textual patterns and a number of dimensions that allow engineers to state their duration, subject, context, decision making, and phase. Finally, the authors formalize: (1) a pattern for selecting the most satisfactory service provider, (2) and another pattern for a portability from a service provider to another. In the same vein, Paschke et al. [46] define a language for designing patterns in the field of decentralized bargaining coordination which is called NPL. Besides, they offer an on-line catalog of design patterns which is not currently operating. Oluyomi, in his doctoral thesis [44], offers a detailed classification scheme for design patterns in agent-based software that includes, among others, negotiation protocols.

Software design patterns are a formalization (via patterns) of a concept (a solution to a software problem) shared (among different developers). Another field that aims to provide formalizations of shared concepts are ontologies. Within the scope of the agreement technologies, Tamma et al. [59] proposed negotiation protocols expressed through a shared ontology. This ontology aims to be general enough to allow developers to cover the entire spectrum of possible negotiation protocols. The main advantage of this approach is that it reduces the need for hard-code in agents, although it is not removed. These agents are supposed to search in the ontology the information required to fulfil the agreement such as the negotiation protocol or the rules for deciding whether the agreement has been reached. In this line of ontologies definition for negotiation, Duran and Churches [22] formalize auction based negotiations combining patterns definitions with the use of ontologies.

Benyoucef and Keller [14] performed an evaluation of several formalisms that can be applied to describe negotiations: natural language agent coordination language, finite state machines, the state diagram, etcetera. The authors conclude that the ideal representation is the state diagrams. However, the above alternatives (agreement patterns, ontology, or a combination of both) are not explored by the authors. Examining these works, the reader can observe the remarkable progress made in describing, cataloguing and publishing agreement patterns. Nonetheless, these efforts contrast with the lack of tools to assess these agreement technologies and to decide which is the best suited pattern in specific domains and instances of these domains. In other words, it is difficult to find methods to know which agreement technology is more fitting for a specific intelligent environment with specific users.

The most relevant research streams in agreement technologies include:

- *Auction theory:* it analyzes protocols and agents' strategies in auctions. Auctions are usually used in systems where the auctioneer wants to sell an item and get the highest possible payment.
- *Negotiation or Bargaining Theory:* the agreement is modeled as a sequential game where agents alternate in making offers according to an underlying protocol.
- *Contracting theory:* a very well-know protocol in this domain is the contract net protocol [57] which allows a contractor agent to contract one or more participant agents to undertake some task.
- *Social Choice Theory:* combining individual preferences, interests, or welfares to reach a collective decision or social welfare in some sense [10].

Among these streams, the authors consider the use of social choice as the most suitable option for resolving conflicts in AmI. The main reason is that it is focused on maximizing social welfare. Furthermore, there are a number of scenarios where users have a peer to peer relationship and, besides expressing their personal preferences, there is nothing else to be said in a negotiation. In contrast, the bargaining theory is also a feasible option for some scenarios, e.g. if agents' preferences may change by argumentation.

2.2. Social choice related works

As explained by Procaccia [47], social choice theory has seen few applications to date. The reason given by the author is that political elections, which are perhaps the most prominent social decision making mechanisms, are very difficult to change. Social choice research has been mainly theoretical, being the work by Arrow et al. [10] its maximum exponent. This research line focuses on verifying that a voting system satisfies certain mathematical properties such as the majority criterion; i.e. if one candidate is preferred by a majority (more than 50%) of voters, then that candidate must win.

In this theoretic line, García-Lapresta et al. [26] define the Borda count in a linguistic decision making context. The authors propose to take into account all the agents' opinions or only the favorable ones for each alternative when compared with each other. Both possibilities are analyzed to find out their properties. Other authors focused on just one property,

proposed or not by them. In this vein, Alcantud et al. [9] proposes the approval consensus measure to quantify the cohesiveness that approval voting situations conveys. Another voting application is the design of fault tolerant systems. In this manner, Linda and Manic [37] improve the handling of uncertain assumptions about the distributions of noisy and erroneous inputs for designing fuzzy voting schemes.

As seen in the introduction, when social choice is used to resolve to conflicts in shared resources into AmI environments these theoretical studies do not respond to a series of questions of great interest such as how much satisfaction should developers expect after including these techniques. To answer these questions, it is necessary a simulation-based experimental research.

Some works simulate different voting systems to study their weaknesses to tactical voting, i.e. strategic vulnerability. Balinski and Laraki [12] performed an experimental investigation by using a set of simulated elections based on the results from a poll of the 2007 French presidential election. The conclusion was that range voting presented the worst strategic vulnerability, while their proposed system, majority judgement, had the best.

Another interesting related work is the tool *politicalsim*[1] which is meant to test different voting systems. Unfortunately, the download links do not work currently. More importantly, the political elections are significantly different from the social choice application this paper deals with. For example, politicalsim only allows 2 to 16 candidates to compete for 1 to 7 seats and limiting shared services to less than 16 configurations is too restrictive.

Aseere et al. [11] present one of the few works which: (1) combines social choice with an practical application distinct from political elections; and which (2) gives experimental results to quantify the benefits obtained. These authors propose a multi-agent system based on an iterative voting protocol where student agents could vote to decide which courses the university would be running.

To the best of authors' knowledge, this paper is the first work which proposes the use of social choice and agent-based social simulation to improve the access to shared services in AmI environments. Given this scope, different simulators focused on testing AmI systems, before or during its deployment, have been reviewed in the following section.

2.3. AmI simulation platforms

A number of approaches dealing with the evaluation of AmI systems by simulated environments can be found in the literature. A well-known tool is Ubiwise [13]. Ubiwise focuses on the use and analysis of environment models for ubiquitous computing systems to be conducted. For this purpose, sensors and its communications can be defined. People are considered individually on a simulation engine based on Quake II, where real people, acting as players in a game, generate information about their own context, which is captured through simulated sensors. Multiple users can link up to the same server to create interactive ubiquitous computing scenarios. However, the virtual subjects are not autonomous, since they must be controlled by the users. The subjects interact with the environment in order to validate its deployed services.

TATUS [45] is another tool which allows experimentation of adaptive ubiquitous computing systems. It is based on a graphic engine called Half-life. In this case, the main novelty is that multiple systems under test may be connected to the engine. This allow the AmI system under test to be adaptive: it changes its behavior in reaction to user's movements and behavior. Other environmental factors such as network conditions, ambient noise or social setting can also be considered. In this manner, some virtual subjects could behave autonomously due to simple AI scripts, although the tool is mainly focused on user-controlled characters and their interactions with the system under test.

Another interesting tool is UbiREAL [42] which allows users to intuitively gain insights into how devices are controlled depending on temporal variation of contexts in a virtual space. The main contribution of UbiREAL is that it simulates physical quantities (e.g. temperature or humidity) and includes a network simulator. This network simulator allows communication between virtual devices and real devices. This is a very important aspect since it is possible to inject reality into the simulations. The behavior of the virtual subjects must be preconfigured to define a route and some actions to perform.

Tang et al. [60] propose a user-centered design methodology that combines the notion of situation (a description of the states of relevant entities) with an application model. Additionally, they offer a domain-specific design language and a set of graphical toolkits covering the life-cycle development of a pervasive application to support the methodology. OpenSim[2] is the simulator employed and each entity is represented as a plug-in module. Users in the simulator are avatars and, therefore, they must be controlled by humans. Nevertheless, by recording context traces, experiments can be repeated without human intervention. A middleware is used to give support for executing the application, including the discovery of available services.

UbikSim [6] is a framework to test AmI systems by agent based social simulation that has been employed in this paper to create the tool that complements it, VoteSim [8]. The main reason is that, unlike the alternatives studied above except OpenSim, UbikSim is open-source and available on-line. Moreover, as explained in Section 5, UbikSim allows developers to model realistic environments with shared services in an easy and graphical manner and to describe automatic users behaviors in a number of formalisms [28].

[1] Politicalsim website: http://politicalsim.com/.

[2] Open Simulator website: http://www.opensimulator.org.

3. Target system, a generic agreement service for shared resources

This sections details the "real world" phenomenon to research on. This is called the *target system* in ABSS terminology [30]. As explained in the introduction, this target system is a generic agreement service for shared resources in intelligent environments.

3.1. Target description

The generic agreement service for shared resources in intelligent environments is assumed to be a *multi-agent system* [62] for several reasons: (1) the agent theory has covered a great variety of negotiation aspects [33] and, therefore, these systems are very appropriate for an agreement service; (2) this paradigm has been widely used for the development of AmI given the *Agent-based Ubiquitous Computing* [40]; and, since agent based simulations are employed to model the target, there is a clear correspondence between the target and the model [50].

Fig. 1 summarizes this target as a multi-agent system. Basically, there are a number of users which can use one of several shared resources in the environment and an agent community which aims to maximize the satisfaction of users.

Each user has assigned an agent, *user agent* (UA), which negotiates on her behalf. The basic elements needed for this are: (1) the preferences of the user with regard to a service, which allow the UA to obtain what the user wants; and (2) the agreement strategies (or negotiation strategies), which allow the UA to make the best out of her participation [14]. Assuming these two basic elements, there are several possible final AmI systems which fit this design. In the most complex case, the user agent detects user location and needs. In the simplest case, the UA can be embedded in a device carried by the user, such as a smartphone, and all data required is given to the agent.

The second agent included in this generic system is the *Agreement Service Agent* (ASA). The ASA provides UAs with the necessary information to negotiate. Firstly, the ASA contains Domain knowledge of services which can be as simple as a list of services with their possible configurations (music themes or TV programs currently available) or an ontology establishing different relations among these configurations. Secondly, the ASA gives the agreement protocols, specified in high-level agent communication languages (ACLs, e.g. FIPA-ACL [23]), which allows UAs to interact independently of the technology employed in their development. These agreement protocols are the negotiation rules, the rules governing the negotiation which have to be shared among negotiating agents regardless of their agreement strategies [14]. Thirdly, the ASA also monitors the negotiations carried out by the UAs and stores the service state (current configuration, current users, etcetera). Finally, the ASA may use the information obtained by monitoring negotiations to elaborate theories about users preferences which can be employed to give a better service [53]. For example, these theories could be used to advance the results of a negotiation or to suggest one of several services according to the preferences similarity among the current users of the service and the incoming user. Note that the ASA is a smart agent according to Nwana classification [43] because its autonomy, cooperation and learning; whereas UAs, assuming that the needs and location learning are not included, are collaborative agents.

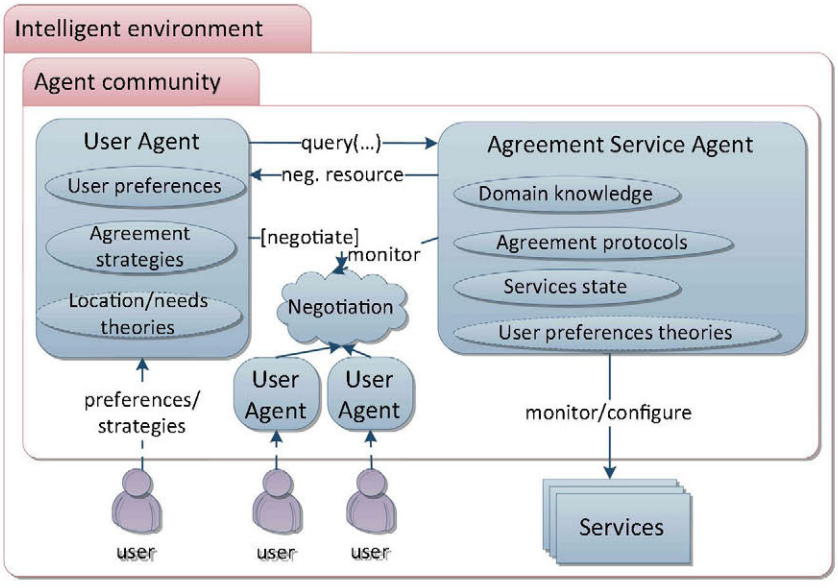

Fig. 1. A general multi-agent based agreement service for intelligent environments.

3.2. Target flow

The flow of the system execution is the following. (1) The user asks her UAs for the services available and their possible configurations and the UA asks the ASA for this information. (2) The user gives its preferences for one service (or several) to the UA and an agreement strategy if wanted. (3) When the user wants to use the aforementioned service, the UA asks about the service state and the ASA returns this information and, optionally, a suggestion based on a users preferences theory (e.g. this is the state of the service requested, but this other is suggested). (4) If the service is being used by other users, the UA requests the agreement protocol necessary to negotiate a configuration for the service with these users' UAs. (5) If the UA agreement strategy does not allow the negotiation to be fulfilled, more information is requested to the user. (6) A negotiation based on this abstract agreement protocol is performed among the UAs. (7) After this negotiation is done, the ASA selects the agreed parameters for the service and, optionally, includes the data exchanged in the observed interactions in its users preferences theory to suggest services to UAs.

3.3. Target justification and motivation

The design of the agent community is inspired by the specialized literature about negotiation in e-commerce. More specifically, it mostly fits the proposal by Benyoucef and Keller [14] where the user gives her preferences and strategies to the agent so she can actually monitor and vary these elements. Despite having been useful for this target definition, other approaches have been ruled out. Examples of these are the negotiation architectures proposed by: Su et al. [58], where a proxy negotiation server does not allow the user any degree of monitoring or control once the negotiation process begins; Chavez et al. [18], where a negotiation agent embeds a predefined negotiation strategy; and, by Kwon et al. [35], where negotiation strategies are public. Negotiation strategies, such as: "I like all configurations C, but since I like $c \in C$ the most, I will deny any alternative to c" should be monitorable, configurable and private.

4. Modeling the target studied

The contribution of this paper is the use of ABSS techniques to evaluate the kind of target explained above, and more specifically, the possible social choice methods employed to carry out a negotiation of shared services. As explained above, a considerable advantage of using ABSS for that purpose is that the simulation of the agent community can be transformed into the final multi-agent system directly or this final system may be considered at the beginning in the simulation [29]. This section aims of creating a model of the target simpler to study than the target itself but descriptive enough to obtain conclusions about the effect of different social choice methods.

4.1. User agent model

This section describes the main considerations in the development of a model for the *user agent* (UA) which is introduced in Section 3 and illustrated in Fig. 1. This includes a description of user's preferences model, the location and need theories conducted by the agent, and the possible agreement strategies.

4.1.1. User's preferences and global satisfaction function

This section describes, on the one hand, the user's preferences model and, on the other hand, the satisfaction metrics that are employed to measure the benefits of using a specific voting algorithm in a specific AmI system.

Definition 1 (*User's preferences*). Given a specific kind of service or a set of services $S = \{s_1, s_2, \ldots s_m\}$, with $m = |S|$, in a intelligent environment, each one of these services have a set of possible configurations C_S, with $n = |C_S|$, which are similar $\forall s \in S$. Let us define user's preferences of a user $u \in U$ for the set of services S at time $t \in \mathbb{R}$ as:

$$p(u,S,t) = < c_1/p_1, c_2/p_2, \ldots c_n/p_n > \tag{1}$$

where: $c_i \in C_S, 1 \leqslant i \leqslant n, p_i \in \mathbb{Z}$, and $0 \leqslant p_i \leqslant 10$. That is, user's preferences are described as a list with considered configurations labeled with a score from 0 to 10. Let us consider the list ordered from high to low preference, i.e. $10 \geqslant p_i \geqslant p_{i+1} \geqslant 0\ \forall i$.

Let us also assume that $p(u,S,t).c_i$ obtains p_i, i.e. the score for c_i, and $p(u,S,t).i$ obtains the configuration c_i.

The specialized literature usually uses some variant of the formal model described by Shoham and Leyton-Brown [56] where $a : o_1 \succ o_2 \succ o_3$ specifies that agent a prefers the outcome (or alternative, or candidate) o_1 to o_2, and the latter to o_3. Nevertheless, this paper labels each outcome, *configuration* in this case of social choice application, with a specific score in order to map easily a voting result with a quantified social welfare as discussed below in this section. Another advantage of the formal model presented in this paper is that it makes trivial for the agents to fulfil the social choice function for a number of voting systems.

Definition 2 (*Social choice function*). A social choice function is a function [56]:

$$C : L^n \rightarrow O \tag{2}$$

where n is the number of agents considered, O denotes a finite set of candidates, and L is the set of non-strict preference ordering total orders (such as $o_1 \succ o_2 \succ o_3$).

Therefore, the social choice function takes the users' preferences as input and returns the chosen candidate.

The social choice function defined above uses the candidates ordered by preferences as input. Implementing this function is feasible for Plurality or Borda voting, for example. On the other hand, the scores for each candidate are needed for other systems such as Range voting and Approval voting as explained in Section 4.2.1. Therefore, with the new preferences model proposed in this paper (see Definition 1), a new social choice function will be defined to cover all voting systems considered (see Section 4.2.1.6).

The configurations set C_S (or O in the previous definition) can have very different complexities as Section 4.2.2 explains. Besides, establishing a comprehensive list of configurations is not always possible. Thus, let us consider that if a configuration $c \in C_S$ is not listed in $p(u,S,t)$, then $p(u,S,t).c = 0$. This is a significant difference between the classical case, political elections, where candidates are well defined. Note that preferences are parametrized by time since they can be dynamic, i.e. users can change the scores given to each configuration throughout time.

For the model presented in this paper, the specific scores can be real preferences given by users and which can be automatically extracted by using social network analysis [17,52], be fixed to model concrete scenarios (as Section 6 considers), or simply be random.

Besides the aforementioned preferences, a number of definitions are detailed here to reach the basic metrics this paper is interested in. The first definition calculates user's satisfaction when using a service at a certain time.

Definition 3 (*User's satisfaction*). Let us define the user's satisfaction for $u \in U_S$ using a service $s \in S$ with a configuration $c \in C_s$ at a time $t \in \mathbb{R}$ as:

$$sas(u,s,c,t) = p(u,S,t).c \tag{3}$$

That is, the satisfaction produced with a configuration is equivalent to the user's preference for this configuration. Therefore, this function is bounded on $[0,10]$.

Realize that this specific definition makes that range voting, a specific voting system explained in Section 4.2.1, obtains always the maximum satisfaction if the group voting is the same. However, as we will see, there are other metrics and factors to be considered which justify the use of different voting methods. As explained in Section 4.2.4, the results of a basic range voting can be improved, for example, by suggesting intelligently the best group to vote.

The state of a service is required to calculate the satisfaction of users accessing that service.

Definition 4 (*State of service*). Let us define a state of service function $state(s \in S, t \in \mathbb{R})$ that, given a service, and a time, returns $< c, U_s >$ where $c \in C_S$ is the configuration taken in s and $U_s \subseteq S$ are the users accessing the service s of type S.

Let us assume that the service configuration is obtained with $state(s,t).c$ and the users set using it with $state(s,t).U_s$.

From this definition, the service satisfaction can be defined.

Definition 5 (*Service satisfaction*). The service satisfaction when used with a configuration $c \in C_S$ can be defined as follows:

$$servSas(s,c,t) = \sum_{u \in state(s,t).U_S} sas(u,s,c,t)/(10 * |state(s,t).U_S|) \tag{4}$$

where $s \in S$ and $t \in \mathbb{R}$. That is, the sum of users satisfaction considering users which are using the service at t. The function is bounded on $[0,1]$.

Given the last definition, the voting results for all services can be integrated to define the global satisfaction.

Definition 6 (*Global satisfaction*). Let us define the global satisfaction as follows:

$$globalServSas(S,t) = \sum_{s \in S} servSas(s, state(s,t).c, t)/|S| \tag{5}$$

where S is the set of services and $t \in \mathbb{R}$.

Finally, the first metric proposed in this paper to quantify the benefits of using voting systems in AmI environments is the following.

Definition 7 (*Accumulated satisfaction (as)*). The accumulated satisfaction for an instant t is:

$$accSas(S,t) = \begin{cases} 0 & \text{if } t = 0 \\ \left(\sum_{i=1}^{t} globalServSas(S,i) \right) \Big/ t & \text{if } t > 0 \end{cases} \tag{6}$$

where S is the set of services and $t \in \mathbb{R}$.

This final expression allows developers to evaluate the satisfaction obtained with different populations simulated and diverse voting systems.

In addition to this basic metric, two more metrics are considered assuming the concept of wanted configuration.

Definition 8 (*Wanted configuration*). Given some user's preferences $p(u,S,t)$, let us define the set of wanted configurations as:

$$wc(u,S,t) = \{p(u,S,t).1, p(u,S,t).2, p(u,S,t).3\} \tag{7}$$

where $u \in U, s \in S$ and $t \in \mathbb{R}$.

Therefore, a configuration is wanted if it is in the top-three most wanted configurations. This definition assumes that the number of possible configurations for a service is greater than three.

The maximum time that a user has spent without a wanted configuration can be interesting to quantify the effects of a voting system over minorities, i.e. users with unusual preferences. Unlike in political elections, owners of services are probably interested in avoiding long waits to get a wanted configuration.

Definition 9 (*Maximum time without wanted configuration (mwc)*). It is defined as a function $maxWithoutWC(S,t)$ which returns a time t_2 with the maximum time that a user has needed to get a wanted configuration in a service of kind S. Where t and t_2 are time instants; and $t_2 \leqslant t$.

Finally, as a third metric, the users with wanted configurations can be defined as follows:

Definition 10 (*Users with wanted configurations (uwc)*). It is defined as a function $usersWithWC(S,t)$ which returns a number in $0 \leqslant \mathbb{Z} \leqslant |U| * t$ with the sum of users that have enjoyed a wanted configuration in a service of type S from the first instant to t. Where U is the set of possible users of a type of service S and $t \in \mathbb{R}$.

4.1.2. Location and need theories

As explained in Section 3, the target complexity strongly depends on how the UA obtains data about user's location and needs. In the most complex case, the user agent has to detect this information. Regarding indoor location, there are a number of works which deal with this problem by using wireless technologies such as Global Positioning System (GPS), Radio Frequency IDentification (RFID), wireless local area network (WLAN), mobile cellular network, bluetooth, etcetera [38]. Identifying the different users and their locations is very challenging, but the problem is even more complicated if the user does not give her service preferences. Assuming that the configuration given for the service is the preferred by the user and conducting learning algorithms is feasible [48]. On the other hand, if this service is shared, users may not like the parameters given. Therefore, sentiment analysis and detecting emotions through face recognition techniques would be necessary in this case.

A well known principle of social simulation, not only in general [30] but also in its application to validate AmI systems [50], is that models have to be as simple as possible but, simultaneously, be descriptive of the reality being modeled. Otherwise, the amount of variables makes unfeasible to conduct a correct analysis and, therefore, to obtain reliable conclusions about the target. Since this paper is focused on evaluating voting systems, for the sake of simplicity, the UA is considered integrated into a smartphone which is carried by the user. The user introduces her preferences and agreement strategies for each service and they are privately stored by the agent. However, although not considered in this paper, including an indoor location system in the model would allow developers of such a system to explore the satisfaction loss when these predictions fail.

4.1.3. Agreement strategies

As explained in Section 2, social choice is just one of the possible ways of reaching an agreement. When implementing argumentations or auctions, for example, there are a number of decisions that must be automatized in the UA: how to reply other agents' arguments, when to bid and when stop bidding, etcetera. In contrast, when considering voting, it could seem that there is no option but simply voting for your favorite candidates according to users' preferences.

Quite the contrary, one the main research streams in social choice theory is the study of *strategic voting* (or tactic voting). Strategic voting is to vote in a manner that contradicts voter's preferences to manipulate the elections global result. If a voting procedure sometimes allows a voter to secure a preferred outcome by voting in a way that contradicts her true preferences, the procedure is said to be susceptible to strategic manipulation. Most voting systems involving three or more candidates, configurations in the case presented in this paper, are susceptible to strategic manipulation [10].

Example 1 (*Tactical voting example*). A good and realistic example of strategic voting is given by Wooldrige [62] in the scope of UK general elections. According to the author, the three most influential parties in UK are: Labor (left-wing party), Liberal (center-left) and Conservative (right wing). Now consider a very left-wing voter in a district with a bias towards the Conservative party. With voting system employed, the plurality voting (see Section 4.2.1), the aforementioned voter could decide that her honest option, the Labor party, will waste a vote and, therefore, it is better to vote the Liberal party. In Spain, exactly the same scenario could be described with the parties: IU (left-wing), PSOE (center-left) and PP (right wing).

For the sake of minimalism, this model assumes trustworthy voters. As in Section 4.1.2, the possible inclusion of tactical voting points out both an interesting extension of this work, and a difference between the application field exposed here and the political elections. Unlike political elections, voting occurs much more frequently to decide shared services configurations. Besides, there is an agent observer, the ASA, which knows the votes produced by different UAs. Therefore, *reputation* research [54] could be applied to evaluate the trustworthiness and reliability of individuals in the multi-agent system.

Finally, even assuming honest users and a UA with all configurations labeled with a score, some voting systems require some strategy to vote. These are discussed in the scope of the different voting systems in Section 4.2.1.

4.2. Agreement service agent

This section describes the main considerations in the development of a model for the *agreement service agent* (ASA) which is introduced in Section 3 and illustrated in Fig. 1. This includes a description of possible agreement protocols, the domain knowledge protocol, the service state, and user preferences theories.

4.2.1. Agreement protocols: voting systems

One of the main functions of the ASA is to give the agreement protocols, voting systems in this paper, to the UAs to allow them to reach an agreement. These protocols are modeled in this paper by using the agreement pattern specified in Fig. 2. The figure shows a reasoning diagram following CommonKads notation [16]: knowledge roles are displayed as squares and inferences as ovals.

In the figure, UAs hold a *voting function* which, from a *vote format* given by the *ASA* and user's preferences, produces a vote. This vote is given to a *social choice function* and it produces a result with the configuration selected. Although not used in this paper, it is interesting to note that there are iterative voting approaches which require to repeat voting several times. This repetition may just be required to break ties.

This agreement pattern, which attempts to cover all voting systems, has a significant difference with formal frameworks presented by Shoham and Leyton-Brown [56] or by Woolridge [62]. In these works, the function receives a preference ordering where $a : o_1 \succ o_2 \succ o_3$ specifies that agent a prefers the outcome (or alternative, or candidate) o_1 to o_2, and the latter to o_3. Although this input is valid for a number of voting systems such as Borda voting or plurality voting; these orders have just not enough information to perform other voting systems such as Approval, Range or Cumulative voting (see their definitions in this section). Therefore, a previous function to generate the vote is required. This function may require a strategy to generate the vote even if agents are completely honest in their preferences. The rest of this section defines different voting systems (including the vote format and voting functions proposed for this model) and a common social choice function.

4.2.1.1. Plurality voting. The Plurality Voting is described as follows:

Definition 11 (*Plurality Voting (PV)*). Each voter casts a single vote and the candidate with the most votes is selected [56].

Therefore, the PV vote format can be defined as follows:

Definition 12 (*Vote format for PV*).

$$vote_{PV} = < c_1/v_1, c_2/v_2, \ldots c_n/v_n > \tag{8}$$

where: $c_i \in C_S, 1 \leqslant i \leqslant n, n = |C_S|, v_i \in \mathbb{Z}, 0 \leqslant v_i \leqslant 1$, and $\sum_{i=1}^{n} v_i = 1$. Therefore, a PV vote is given as a vector of configurations where there is just one configuration labeled with 1, the favorite configuration, and the remaining candidates are labeled with 0. Let us also assume that $Vote_{PV}.c_i$ returns v_i.

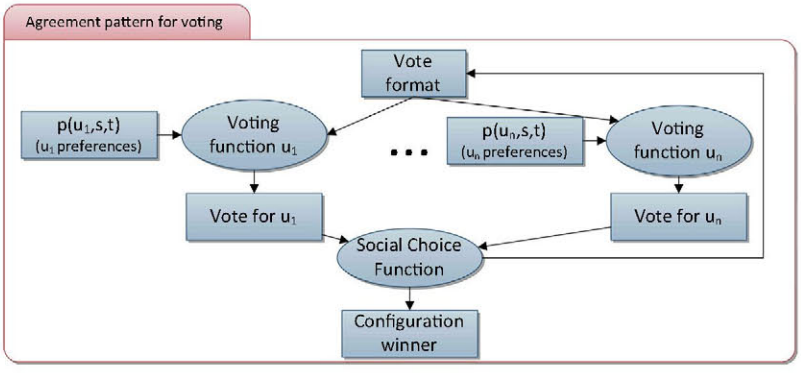

Fig. 2. General agreement patter for voting.

```
1 vote_PV function voting_PV(u,S,t) {
2 vote_PV =< c1/0, c2/0, ...cn/0 >
3 Configuration c = p(u,S,t).1
4 vote_PV.c = 1
5 return vote_PV
6 }//end function
```

Fig. 3. A formal voting function to vote with plurality voting system.

Although this format is not the most efficient for this specific voting system, as shown below, it can be easily extended to cover the remaining systems. Besides, this format allows clustering algorithm to be applied directly considering votes as tuples (see Section 4.2.4). In this case, there is a clear mapping from the preferences and the vote format to the vote. Only the first configuration in the preferences, see Definition 1, is 1 in the vote. This *voting function* for PV is given as an algorithm in Fig. 3. Observe that $p(u,S,t).1$ returns the most preferred configuration (see Definition 1). Note also that, assuming an honest agent, there is just one manner of defining the voting function.

4.2.1.2. Borda voting. The Borda Voting is described as follows:

Definition 13 (*Borda Voting (BV)*). Each voter submits a full ordering on the candidates. This ordering contributes points to each candidate; if there are n candidates, it contributes $n-1$ points to the highest ranked candidate, $n-2$ points to the second highest, and so on; it contributes no points to the lowest ranked candidate. The winners are those whose total sum of points from all the voters is maximal [56].

Therefore, the BV vote format can be defined as follows:

Definition 14 (*Vote format for BV*).

$$vote_{BV} = < c_1/v_1, c_2/v_2, \ldots c_n/v_n > \tag{9}$$

where: $c_i \in C_S, 1 \leqslant i \leqslant n, n = |C_S|$, $v_i \in \mathbb{Z}, 0 \leqslant v_i \leqslant n-1$, and $\sum_{i=1}^{n} v_i = \frac{n(n-1)}{2}$. Therefore, a BV vote is given as a vector of configurations labeled with integers from $n-1$ for the most wanted configuration, $n-2$ for the second most wanted, and so on. Let us assume that $Vote_{BV}.c_i$ returns v_i.

Again, there is a clear mapping from the preferences and the vote format to the vote. This *voting function* for BV is given as an algorithm in Fig. 4. Note that, assuming an honest agent, there is just one manner of defining the voting function.

4.2.1.3. Range voting. The Range Voting is described as follows:

Definition 15 (*Range Voting (RV)*). Each voter rates each candidate with a number within a specified range, such as 0 to 10 or 1 to 5. All candidates should be rated.

Therefore, the RV vote format can be defined as follows assuming a score from 0 to 10:

Definition 16 (*Vote format for RV*).

$$vote_{RV} = < c_1/v_1, c_2/v_2, \ldots c_n/v_n > \tag{10}$$

where: $c_i \in C_S, 1 \leqslant i \leqslant n, n = |C_S|$, $v_i \in \mathbb{Z}$, and $0 \leqslant v_i \leqslant 10$. Therefore, a RV vote is given as a vector of configurations labeled with integers from 0 to 10. Let us assume that $Vote_{RV}.c_i$ returns v_i.

```
1  vote_BV function voting_BV(u,S,t) {
2  vote_BV =< c1/0, c2/0, ...cn/0 >
3  integer n = |C_S|
4  for(int i=0;i<|p(u,S,t)|;i++){
5     Configuration c = p(u,S,t).i
6     vote_BV.c = n - 1
7     n--
8  }//end while
9  return vote_BV
10 }//end function
```

Fig. 4. A formal voting function to vote with Borda voting system.

```
1 vote_RV function voting_RV(u,S,t) {
2 vote_RV =< c1/0, c2/0, ...cn/0 >
3 integer n = |C_S|
4 for(int i=0;i<|p(u,S,t)|;i++){
5   Configuration c = p(u,S,t).i
6   vote_RV.c = p(u,S,t).c
7 }//end while
8 return vote_RV
9 }//end function
```

Fig. 5. A formal voting function to vote with range voting system.

Assuming the user's preferences model detailed in Section 4.1.1, there is a clear mapping from the preferences and the vote format to the vote. The *voting function* for RV is given as an algorithm in Fig. 5. Observe that, assuming an honest agent, there is just one manner of defining the voting function. Finally, as explained above, the mere order of preferences does not allow a vote to be calculated. This motivates the user's preference models introduced in Section 4.1.1.

4.2.1.4. Approval voting. The Approval Voting is described as follows:

Definition 17 (*Approval Voting (AV)*). Each voter can cast a single vote for as many of the candidates as she wishes; the candidate with the most votes is selected [56].

Therefore, the AV vote format can be defined as follows:

Definition 18 (*Vote format for AV*).

$$vote_{AV} = < c_1/v_1, c_2/v_2, \ldots c_n/v_n > \tag{11}$$

where: $c_i \in C_S, 1 \leqslant i \leqslant n, n = |C_S|$, $v_i \in \mathbb{Z}$, and $0 \leqslant v_i \leqslant 1$. Therefore, a AV vote is given as a vector of configurations labeled with 0 or 1 without constraints in the number of configurations approved. Let us assume that $Vote_{AV}.c_i$ returns v_i.

Unlike the systems studied above, mapping from the preferences and the vote format to the vote in AV requires some strategy. This *voting function* for AV is given as an algorithm in Fig. 6. Realize that, unlike the systems studied above, two honest agents with same preferences can produce different votes. That is why AV requires a strategy which is in the *isApproved* function. Based on the user's preferences model presented in Section 4.1.1, some intuitive approaches are: (1) approving any configuration rated over 5; (2) or taking just the top-three most wanted configurations (see Definition 8).

4.2.1.5. Cumulative voting. The Cumulative Voting is described as follows:

Definition 19 (*Cumulative voting (CV)*). Each voter is given k votes, which can be cast arbitrarily (e.g., several votes could be cast for one candidate, with the remainder of the votes being distributed across other candidates). The candidate with the most votes is selected [56].

Therefore, CV vote format can be defined as follows:

Definition 20 (*Vote format for CV*).

$$vote_{CV} = < c_1/v_1, c_2/v_2, \ldots c_n/v_n > \tag{12}$$

where: $c_i \in C_S, 1 \leqslant i \leqslant n, n = |C_S|$, $v_i \in \mathbb{Z}, 0 \leqslant v_i \leqslant 1$, and $\sum_{i=1}^{n} v_i = k$. Being k the number of votes allowed by cumulative voting. Therefore, a CV vote is given as a vector of configurations where configurations are labeled with a number bounded on $[0, k]$ and where the labels sum is k. Let us also assume that $Vote_{PV}.c_i$ returns v_i.

```
1 vote_AV function voting_AV(u,S,t) {
2 vote_AV =< c1/0, c2/0, ...cn/0 >
3 for(int i=0;i<|p(u,S,t)|;i++){
4   Configuration c = p(u,S,t).i
5   if(isApproved(u,S,t,c) vote_AV.c = 1
6 }//end while
7 return vote_AV
8 }//end function
```

Fig. 6. A formal voting function to vote with Approval voting system.

Inasmuch as there is not a clear mapping from the preferences and the vote format to the vote, a voting function has to be defined to allow the agent to conduct this mapping. For that purpose, let us define the probability of adding a vote to a configuration in CV.

Definition 21 (*Probability of adding a vote to a configuration in CV*). The probability of $vote_{CV}.c \geqslant 1$, with $c \in C_S$, is defined as follows:

$$prob_{CV}(u,S,t,c) = \frac{p(u,S,t).c}{\sum_{i=1}^{n} p(u,S,t).c_i} \tag{13}$$

with $n = |C_S|$. Therefore, the higher is the user's preference for a specific configuration compared to her preferences for other configurations, the higher is the probability of giving one (or more) of the k votes allowed in the former configuration.

Finally, this probability can be used to specify a *voting function* for CV which is given as an algorithm in Fig. 7. This paper experiments with a CV where $k = |C_S|$ and where all users follow the same strategy. This voting function includes a strategy because, as in approval voting and unlike other voting systems, two honest agents with same preferences can produce different votes.

4.2.1.6. Common social choice function. As shown in Fig. 2, the social choice function takes votes and return a winner configuration. As explained, bear in mind that this social choice traditionally takes preferences orders instead of votes. In contrast, the votes formats and vote functions defined above allow the model to have a simple and common social choice function for all voting systems.

Firstly, the sum of votes has to be obtained.

Definition 22 (*Sum of votes*). Assuming a set V of votes

$$vote_{xj} = < c_1/v_1, c_2/v_2, \ldots c_n/v_n >$$

being x some voting system and j the user holding the voting function, the sum of votes is defined as follows:

$$sumOfVotes_x(V) = < c_1 \Big/ \sum_{j=1}^{j<|V|} vote_{xj}.c_1, \ldots c_n \Big/ \sum_{j=1}^{j<|V|} vote_{xj}.c_n \tag{14}$$

where $c_i \in C_S, 1 \leqslant i \leqslant n$, and $n = |C_S|$. Let us assume that $sumOfVotes_x.c_i$ returns $v_i \in \mathbb{Z}$, the final score for configuration c_i.

The common social choice can be defined employing the sum of votes:

Definition 23 (*Common social choice function*). The social choice function for a set of votes V in some voting system x is defined as:

$$SC_x(V) = \{c \in C_S : sumOfVotes_x.c = max(sumOfVotes_x.c_i) \forall c_i \in C_S\} \tag{15}$$

where: $c_i \in C_S, 1 \leqslant i \leqslant n$, and $n = |C_S|$.

```
1  vote_CV function voting_CV(u, S, t)
2  vote_CV =< c_1/0, c_2/0, ...c_n/0 >
3  integer k = votes allowed
4  integer vg= 0; //votes given
5  while(vg ≠ k){
6    for each c_i in C_S{
7      r = random float
8      if(r ≤ prob_CV(u, S, t, c_i) ){
9        vote_CV(u, S, t).c_i++
10       vg++
11     }//end if
12   }//end for
13 }//end while
14 return vote_CV
15 }//end function
```

Fig. 7. A formal voting function to vote with cumulative voting system.

Realize that several configurations may obtain the same score after adding the votes. Ties must be broken according to a tie-breaking rule (e.g. based on a lexicographic ordering of the candidates) [56]. For the sake of having a simple model, the tie-breaking rule used is the random selection among the configurations ended in a tie.

4.2.2. Domain knowledge

The ASA needs not only to provide the UA with an agreement protocol, but also to give valid knowledge structures to fulfil the protocol when necessary. In this scope, this knowledge basically includes the possible configurations in C_S for a specific kind of service S.

In the most simple case, there is a limited and simple list of possible configurations. For example, to select a music theme, a list of all available themes could be offered. This basic case could be expanded considering more service parameters such as the volume for the music case, or the singer of the theme if it has several versions. Therefore, a UA could vote for the following configuration $c \in C_S$:

$$(theme = \text{“}my\ way\text{”}) \wedge (volume \leqslant 10) \wedge (singer = \text{“}Frank\ Sinatra\text{”})$$

Although not employed in this paper, this kind of complex configurations raises another interesting issue uncovered in social choice literature. When voting a political candidate, specifying the kind of behavior the voter is supporting is not possible. But in the services configuration case, there can be unlimited candidates slightly different. In this example, the voting system could isolate the configuration terms and vote iteratively. That is: voting for the theme, then voting for the volume for the winner of the theme, and so on. However, a user could consider that the specific theme is only acceptable if the singer is the one she prefers. Another alternative could be enumerating all configurations suggested by users and vote for each one of them as a whole. Nonetheless, with this solution, two equal songs with different volume would be competing against each other having a vote fragmentation.

4.2.3. Service state

In addition to agreement protocols and domain knowledge, the ASA monitors the services to let the UAs know if a service is being used and with which configuration. In Definition 4 of Section 4.1.1, the state of service was defined as $< c, U_s >$ where $c \in C_S$ is the configuration taken in the service s of type S and $U_s \subseteq U$ are the users accessing the service. From this basic information, the UAs decide if they want to start an agreement protocol in a specific service to change its configuration.

4.2.4. User preferences theories

In the model presented, the ASA holds user preferences theories which can be used to improve intelligently the global satisfaction. More specifically, considering more than one service of a specific type S ($|S| > 1$), the ASA can propose that the UA should vote in the specific service where current users share more preferences with the user represented by the mentioned UA. Therefore, the ASA can provide the UAs with *pre-selection* mechanisms to choose the best group to vote in.

Again, this is a significant difference with political scenario where voters cannot choose the place to vote; i.e. if somebody does not like Spanish candidates, the voter is not allowed to vote in U.K. The following example illustrates the considerable impact that the pre-selection has in global satisfaction.

Example 2 (*Pre-selection group example*). Let us consider four users (from u_1 to u_4) which try to share a kind of service S with two instances ($S = \{s_1, s_2\}$). The users' preferences according to the Definition 1 could be the following:

- $p(u_1, S, t) = p(u_2, S, t) =< c_1/10, c_2/0 >$
- $p(u_3, S, t) = p(u_4, S, t) =< c_2/10, c_1/0 >$

Note that unlike votes, preferences are ordered (see Definition 1). If u_1 and u_2 with a range voting, the service satisfaction for s_1 (see Definition 5) is 1 (100%). On the other hand, if u_1 vote with u_3, this satisfaction is reduced to 0.5. Besides, the results are exactly the same if considered a number of voting systems such as Plurality, Borda, Approval or Cumulative voting.

This example illustrates that a good pre-selection policy not only has impact in global satisfaction, but also can be can be more important than the voting system used. This important fact is ignored by social choice literature because this problem does not occur in political elections.

Modeling correctly these policies involves considering the kind of information that the ASA has available. Ideally, this agent would have access to users' preferences as UAs. On the contrary, the ASA only can try to infer them based on the interactions observed. It is important to note that depending on the voting system employed, the information varies extensively. A vote with range voting gives all preferences information assuming an honest agent. In contrast, a vote with approval vote mixes the most wanted configurations with the just approved.

In this scenario, the use of machine learning techniques to infer automatically users' preferences is very intuitive [53]55. Regardless the voting system used, the information perceived are votes in the format specified in Section 4.2.1: vectors of integers where each position represents a configuration service. As a result, the use of distance functions and clustering approaches is straightforward. *Cluster analysis* is the partitioning, or more generally covering, of a data set into groups, also called clusters, so that data points within a group are similar to each other in some sense, typically according to a chosen

distance measure [41]. In the use of clustering presented in this paper, the last votes known by the ASA for a service are the data set and the different clusters represent available services. This means that the number of clusters, whose estimation is one of the most challenging problems of clustering [36], is a well known number. Furthermore, distance functions can be used as greedy alternatives to clustering, i.e. proposing a user to vote for the service where the sum of distances between her last known vote and the current users' known votes is the least. Some possible distances and clustering techniques which are suitable to address this issue are:

- *Euclidian distance* (or ordinary distance). The Euclidian distance between an instance $a1$ with n attribute with values $a1_1, a1_2, \ldots a1_n$ and an instance $a2$ with values $a2_1, a_2, \ldots a2_n$ is defined as:

$$\sqrt{(a1_1 - a2_1)^2 + (a1_2 - a2_2)^2 \ldots + (a1_n - a2_n)^2}$$

- *Manhattan distance* (or city block distance). The Manhattan distance between an instance $a1$ with n attribute with values $a1_1, a1_2, \ldots a1_n$ and an instance $a2$ with values $a2_1, a_2, \ldots a2_n$ is defined as:

$$\sqrt{|a1_1 - a2_1| + |a1_2 - a2_2| \ldots + |a1_n - a2_n|}$$

- *K-means*. This is the most classic clustering technique. It is aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean [39].
- *Expectation–maximization (EM) algorithm*. An iterative method for finding maximum likelihood or maximum a posteriori estimates of parameters in statistical models, where the model depends on unobserved latent variables [20].

These well known methods are not detailed since they have not been modified or extended for its application in this paper as, for example, the voting systems presented in Section 4.2.1. Moreover, although there are countless distance function and clustering approaches which are fitting, the goal of this paper is not to compare them all but demonstrating the consequential effect that these algorithms can cause when used in the social choice scope.

5. Implementing an agent based social simulation for the model proposed

The implementation of the model detailed in Section 4 is presented in this section. Beyond giving a hard coded implementation for the experiments presented in this paper, a complete tool to evaluate social choice techniques into intelligent environments by agent based social simulation is provided. The authors strongly believe that free and open-source code is a fundamental complement to research works in computer sciences because it allows readers to verify the results and to extend them.

The tool is a *UbikSim* [6] library called *VoteSim* [8]. VoteSim is conceived not only to guarantee the reproducibility of the results presented in this paper, but also to be extended and adapted by developers. Fig. 8 shows VoteSim GUI for the case study of Section 6. VoteSim gives a framework: to asses different voting systems or new agreement patterns to be included in the tool; to study the effect of more sophisticated machine learning techniques that the introduced in Section 4.2.4; to experiment with different users behaviors in accessing shared resources and in voting; and, to experiment with different environments and services.

VoteSim inherits from UbikSim, introduced in Section 2.3, a number of facilities which assist in the aforementioned tasks. UbikSim [6] is an open-source platform for agent-based social simulation that offers not only a graphical editor to model realistic environments, but tools to define users models immersed in that environment [28]. Agents' behaviors in UbikSim are defined as a hierarchical automaton where transitions from one state to another are governed by probability distribution functions. A number methodologies supported by UbikSim and based on Gilbert and Troitzsch's methodology [30] have been proposed to evaluate Aml applications and the realism of users' models [28] [53].

5.1. VoteSim main classes and execution modes

As UbikSim, VoteSim is fully written in the Java programming language. VoteSim main classes are displayed in Fig. 9 and described as follows. VoteSim includes *Persons* or users employing *UserAgents* (UA) which in turn use an *AgreementServiceAgent* (ASA), as described in the target system explained in Section 3. This last agent provides the UAs with voting methods explained in Section 4.2.1 and implemented in the class *VotingMethod* and its subclasses. Moreover, this UA can offer recommendations based on clustering and distance functions as explained in Section 4.2.4 and implemented in the class *Preselection* and it subclasses. The services to be shared are instances of the class *SharedService*. Finally, a *MonitorService* implements and logs the metrics discussed in Section 4.1.1.

VoteSim has three different execution modes:

- The class *VoteSim* executes a simulation in VoteSim with its default parameters (described in Section 5.2). A random seed can be passed as argument to ensure the reproducibility and repeatability of the simulation. Although this may seem an obvious feature, some popular social simulation frameworks, NetLogo [4] for instance, do not allow specifying the seed for all stochastic processes such as the multi-agent scheduling algorithm.

Fig. 8. VoteSim GUI.

- The class *VoteSimWithGUI* adds a graphical user interface (see Fig. 8) to *VoteSim* to verify and study single simulations. Moreover, the simulation parameters, such as the voting and pre-selection method, can be changed through the GUI.
- The class *VoteSimBatch* is given for the batch processing of VoteSim simulations, i.e. it executes simulations changing their parameters without manual intervention. The main parameters changed in different simulations executions are: the random seed (which is used to generate random and different preferences in users), the voting method, and the pre-selection method.

5.2. Setting up VoteSim and reproducing the experiments

This section details how to set up the experimental framework and how to reproduce exactly the experiments presented in Section 6. Besides giving illustrative guidance to allow others to replicate the proposal application, this section offers technical details to allow the readers to conduct their own usage of the approach based on the VoteSim [8] code released with this paper.

As explained in Section 5, the use of VoteSim [8] includes the UbikSim [6] facilities to model an environment graphically. Fig. 10 shows the 3D display of the simulation. This environment has been equipped with three TVs in the hall with the following possible configurations: $C = \{$*"sports", "sitcom", "documentary", "soap", "cartoon", "news", "reality", "quiz", "movie"*$\}$. The reader can define her own environments or modifying the one provided by using the UbikEditor (class *UbikEditor*). New services can be created by extending the class *SharedService*. The TVs possible configurations also can be modified in the class *TV* (see UML class diagram in Fig. 9).

Fifty users try to access the TVs with random preferences following the format introduced in the Definition 1 of Section 4.1.1. These users follow the next behavioral model. With a probability of 0.05 in each time step, the user: goes to a random position in the hall; goes to a TV without users if possible; otherwise, if the ASA recommends a TV, she goes to use it; if no recommendation is given, the closest TV is chosen; once the TV is chosen, the UAs hold by the TV users start a voting via the ASA to decide the configuration; the user stays watching TV for a random number of time steps bounded on [0, 100]; the user comes back to her room and stays there for a random number of time steps bounded on [0, 100]; and the process is repeated. These behavioral parameters are easily configured in the class *UserAgent*, while the number of users to be simulated is a parameter of the class *VoteSim*.

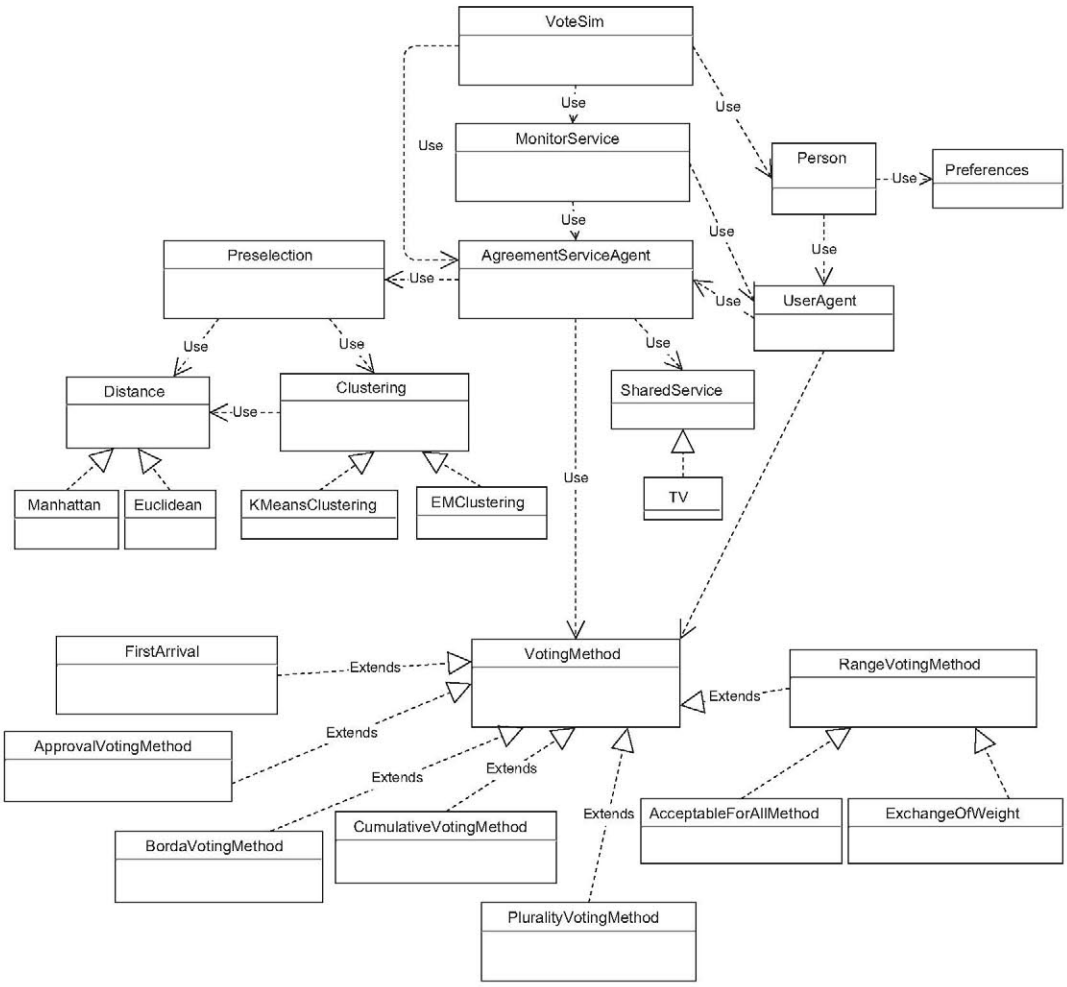

Fig. 9. VoteSim, UML class diagram.

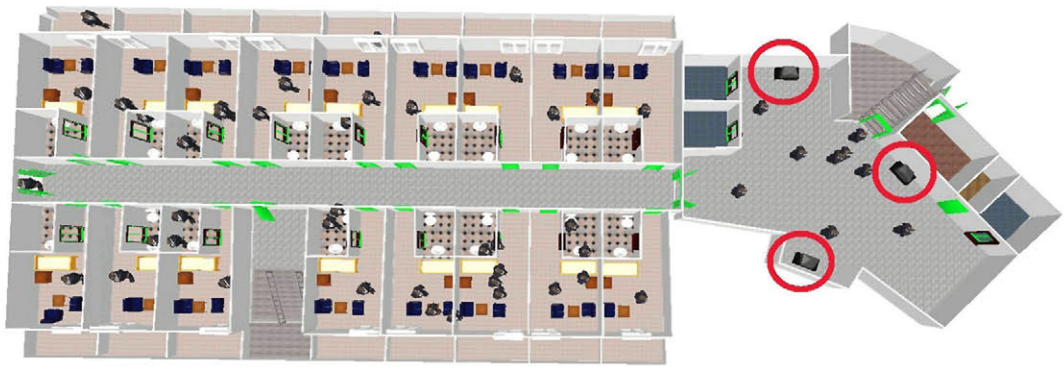

Fig. 10. Hotel floor to evaluate target system.

The basic metric to quantify the suitability of a voting system combined or not with a pre-selection method is the *accumulated satisfaction* (see Definition 7 in Section 4.1.1). Furthermore, the metrics *maximum time without wanted configuration* (Definition 8) and *users with wanted configuration* (Definition 10) are also used. The implementation of these metrics can be revised in the class *MonitorService*. This class can also be used by the user to introduce new metrics.

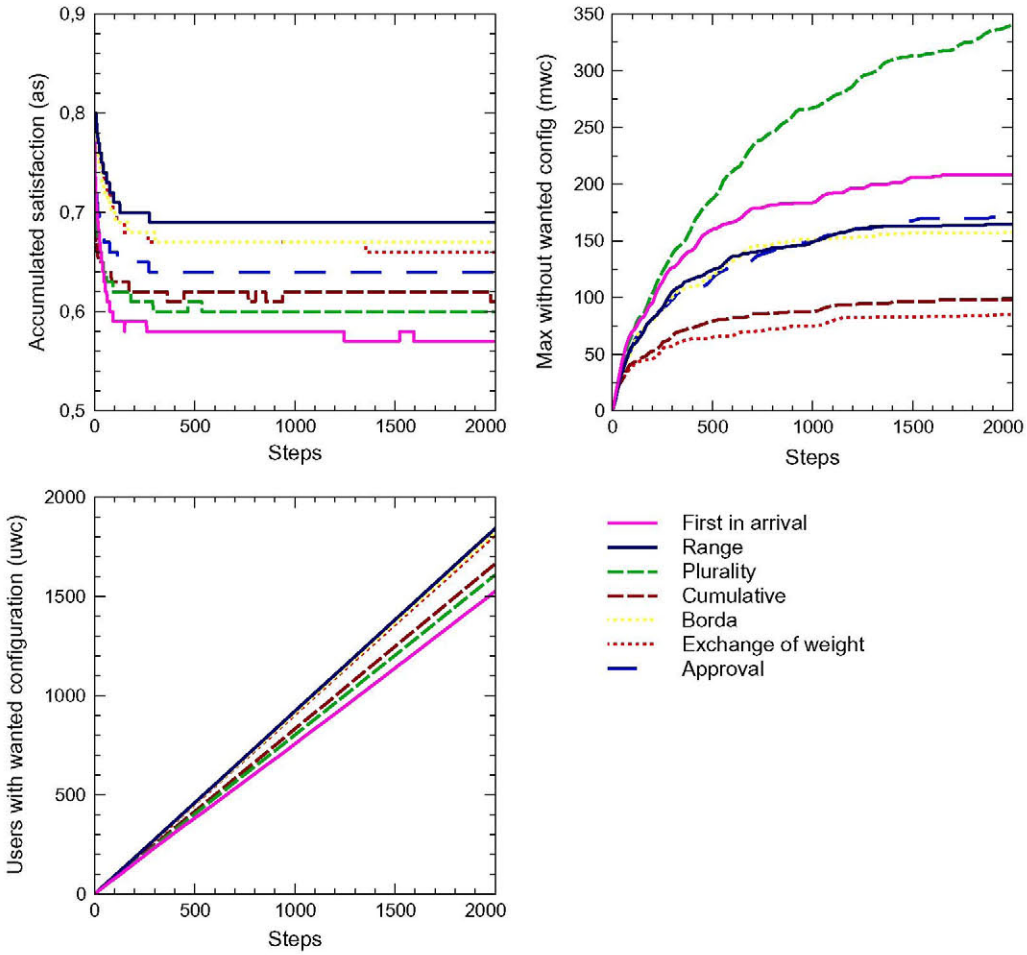

Fig. 11. Experimental results, different voting methods without pre-selection techniques. Tables with experiments results extended are available on-line [2].

The experiments are conducted during 2000 time steps and repeated 100 times with different users populations randomly generated. All combinations of voting systems explained in Section 4.2.1 with pre-selection methods introduced in Section 4.2.4 are considered. These parameters can be configured in the class *VoteSimBatch*.

As a result, to reproduce the experiments presented below, the interested reader just has to: visit the VoteSim [8] website, download it, install it, and execute the *VoteSimBatch* class. Two output files will be produced with the mean and the standard deviation of each metric considered and each combination of voting system with pre-election method for the 100 users populations. Inasmuch as the *"time step"* unit is employed to discuss the time dimension, the specific machine where the simulations have been conducted is irrelevant.

Note that this section has dealt with the possibility of reproducing the entire study by other researchers independently, which is one of the main principles of the scientific method. Another concern would be the variation of outcomes of an experiment carried out in conditions varying within a typical range [3], e.g. measurements are carried out over different users populations (as in the experiments presented). For that purpose, the standard deviation, which is often used to quantify reproducibility [3], is calculated in the experiments (see on-line results [2]). VoteSim also provides researchers with the possibility of including more statistical operations besides the mean and standard deviation in the results (see *VoteSimBatch* class).

6. Case study

As explained in Section 5.2, the case study chosen presents a hotel floor where there is a shared hall with three large television screens that can be used by different clients or users. Fig. 10 shows the floor and the shared services marked in circles. Users have *user agents* (UAs) which know their preferences for the shared service and their location. When more than one client accesses a shared resource simultaneously, UAs contact an *agreement service agent* (ASA) to try to reach an agreement

by some social choice method. This ASA can suggest another service to the UA based on past interactions performed by the latter agent. Once a consensus is reached among different UAs, the ASA selects the chosen configuration. The voting is repeated again when new clients join the group or a client that accesses the resource leaves it.

The main goal of this case study is to quantify the benefits in global satisfaction obtained by using social choice methods and the effects of different pre-selection methods.

6.1. Experimental results on voting methods

The results for different voting methods without pre-selection techniques are shown in Fig. 11. Regarding the accumulated satisfaction (*as*), the range voting offers the best performance: 69%. This is a straightforward result considering the mapping from preferences to the voting function for range voting (see Section 4.2.1). On the other hand, the worst result is with a basis policy included in the experiments: the first user in accessing the resource decides the configuration (the second one decides when this leaves, and so on). This gives an accumulated satisfaction of 57%. Therefore, in this case, voting methods can increase satisfaction by 12%.

One interesting difference between the social choice application considered in this paper and the hegemonic case contemplated in social choice literature, political elections, is that, even giving random preferences, there are significant differences in satisfaction. In political elections and a number of cases, random preferences cause uniform results whatever is decided with whatever method used because there are always plenty of voters happy with the election result. As shown in the experiments, this does not happen when different subsets of users vote when they want to use a shared service.

In addition to the two extreme cases discussed above, the Borda method gets 67%, Approval 65%, Cumulative voting 61%, and Plurality 60%. Therefore, although range voting gets 4% more satisfaction, Approval voting could be employed due to its better usability, i.e. it is easier for users to decide whether a configuration is approved or not than marking each option.

Regarding the number of users with wanted configurations (*uwc*), range voting keeps offering the best result. It is followed by: Approval, Borda voting, Cumulative voting, and Plurality voting. Therefore, Approval is also a good alternative to range voting given its usability.

With respect to the maximum time without wanted configuration (*mwc*), the best results are given by the cumulative voting with 98.1 t.s. (time steps). It is followed by: Borda (157.95), Range (164.75), Approval (171.9), First in arrival (208.3), and Plurality (340.2). Therefore, according to this metric, the plurality method is even worse that not having a voting method since minorities can have extremely long waits to access a service. Besides, this confirms the hypothesis that deciding the best voting method for a case is not trivial.

This paper proposes a new voting method called *exchange of weight* which attempts to get good results in the three metrics employed.

Definition 24 (*Exchange of weight voting (EWV)*). This method is a variation of range voting where after the voting, the configuration selected must meet the following condition: those users whose wanted configurations include the configuration considered (see Definition 8) have to posses enough "negotiation weight" to be given to the remaining users.

Therefore, when considering a configuration c for a kind of service S in an instant t, there is a group of users $u_c \in U_c$ who want c ($p(u_c, S, t).c \in wc(u_c, S, t)$) and other group $u_{\neg c} \in U_{\neg c}$ who do not want it ($p(u_{\neg c}, S, t).c \notin wc(u_{\neg c}, S, t)$). To choose the configuration c, each user $u_{\neg c}$ must receive a weight of $p(u_{\neg c}, S, t).c_1 - p(u_{\neg c}, S, t).c$, i.e. the difference of preference between her favorite configuration c_1 and the configuration c considered. If U_c has not enough negotiation weight to obtain c as chosen configuration, the next most voted configuration according to range voting is considered. If all configurations have been considered, the first most voted configuration according to range voting is considered. All agents start with a weight of 10 so they can convince another user $u_{\neg c}$ with $p(u_{\neg c}, S, t).c = 0$ of using c.

The idea of this method is that users with strange preferences can obtain enough weight after several negotiations to force other users to select the configuration they want. The chart displayed in Fig. 11 shows that this method has the best performance in the *mwc* metric: 85.35 t.s. (being the cumulative voting the second best result with 98.1 t.s.). Simultaneously, the exchange of weight method obtains better results than the cumulative voting for the other two metrics: *as* and *uwc*. However, for these metrics, Range and Borda voting outperform the exchange of weight method slightly (accumulated satisfaction of: 69%, 67%, and 66%, respectively) at the expense of needing almost the double time in *mwc* metric. Therefore, although this novel voting method proposed here is a good balance between the metrics considered, the social choice technique employed will depend on the metric to be maximized. These experiments support the hypothesis that deciding the best social choice algorithm is not trivial and that agent based social simulations are an appropriate tool to make this decision.

6.2. Experimental results on voting methods combined with pre-selection techniques

To study the effect of different pre-selection mechanisms, the two extreme cases (first in arrival and range voting) have been combined with each of the four methods explained in Section 4.2.4. The results are illustrated in Fig. 12. Take into account these experiments are repeated with exactly the same 100 random populations that the previous ones. This reproducibility is a substantial advantage of the simulation-based evaluations.

As shown in the figure, the combination of range voting with a simple pre-selection method based on the Euclidean distance obtains the best satisfaction and number of users with wanted configurations. Even when comparing this to the use of most sophisticated techniques such as the use of k-means. In contrast, EM gets the best result in the *mwc* metric. As seen, this metric is in conflict with the other two.

Note that, to assure reproducibility, these experiments consider the default parameters for EM and k-means in Weka [15] (except the number of clusters, see Section 4.1.1). Weka is an open source collection of machine learning algorithms which is integrated into VoteSim to offer a number of group pre-selection methods.

The satisfaction that range voting presented without pre-selection, 69%, reaches 79% with the use of the euclidean distance described in Section 4.1.1. These experiments confirm the hypothesis that pre-selection methods are as important as the social choice algorithm in this scope.

The Fig. 13 shows the use of the pre-selection method based on the Euclidean distance and its combination with the voting system studied. This pre-selection method has been selected to be displayed since it obtained the best satisfaction. Furthermore, the experiments results with all possible combinations of pre-selections and voting systems are available on-line [2].

In these experiments, the Borda and Exchange of weight voting draw in satisfaction with 78% after range voting with 79%. Besides, it is interesting to note that, including pre-selection, the plurality behaves better than cumulative for *as*. On the other hand, the cumulative voting offers the best mark in *mwc*, even better than the exchange of weights. Besides, this mark is considerably better than the one achieved without pre-selection (27.7 t.s. instead of 98.1 t.s.). Nevertheless, this last voting method proposed in this paper keeps giving an interesting balance between a good *as* (the second best one) and a good *mwc* (the second best one).

These experiments also show that the satisfaction given by first in arrival with the pre-selection, 67%, is almost as good at the satisfaction presented by range voting without pre-selection, 69%, and equal or better that the remaining voting systems

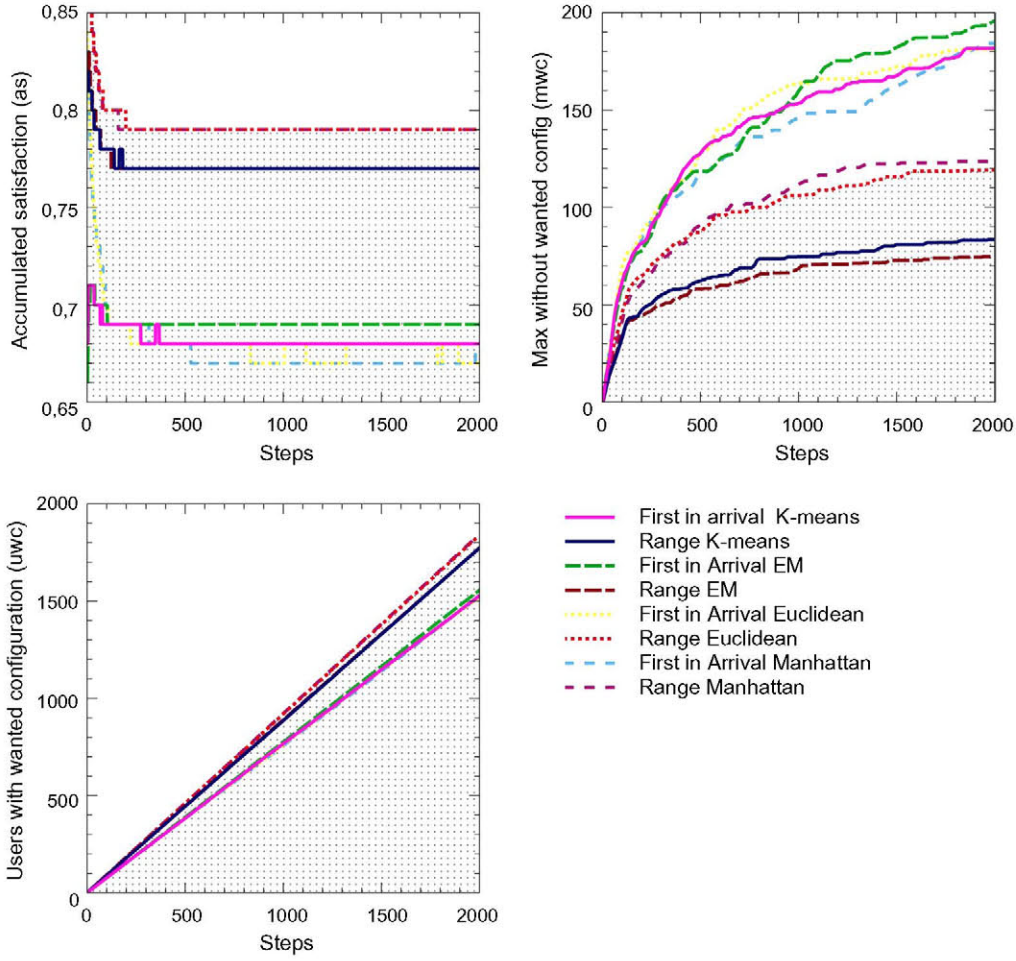

Fig. 12. Experimental results, different group pre-selection methods with first in arrival and range voting. Tables with experiments results extended are available on-line [2].

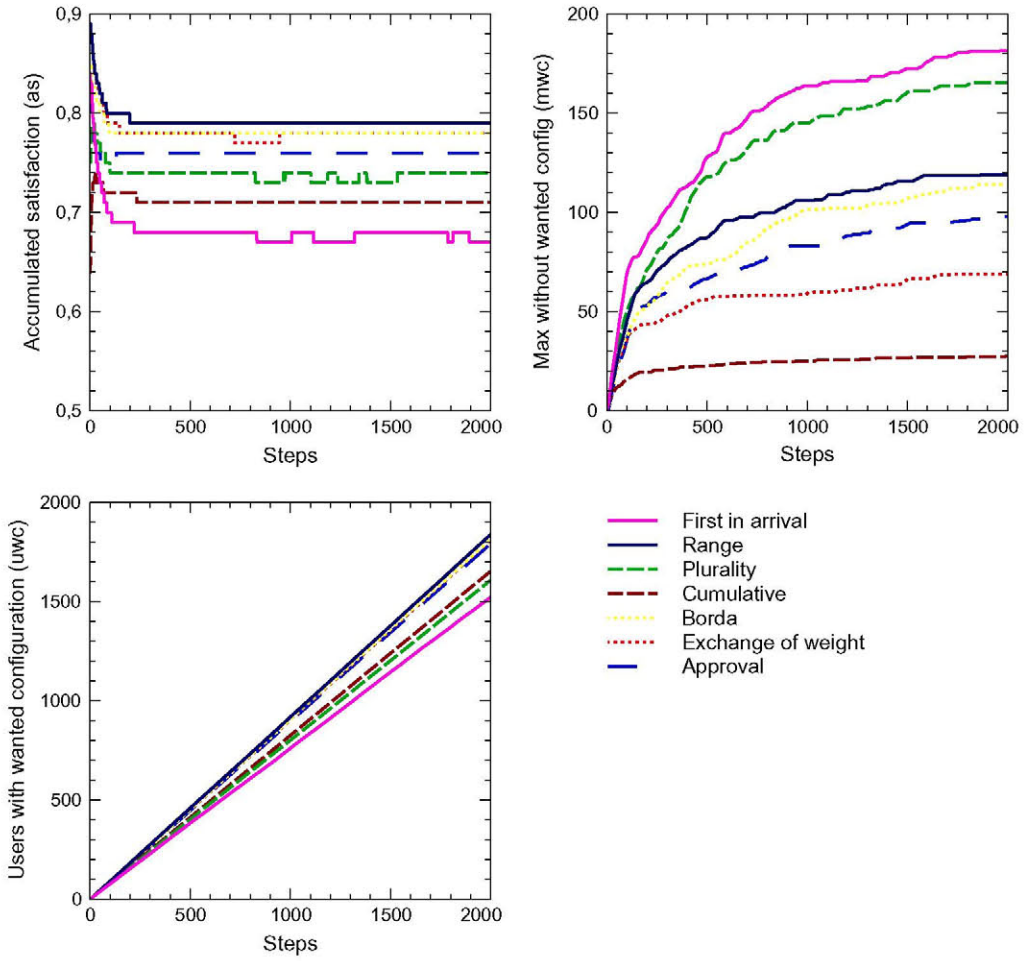

Fig. 13. Experimental results, a pre-selection method based on the Euclidean distance combined with different voting systems. Tables with experiments results extended are available on-line [2].

studied without pre-selection (see Fig. 11). Again, these experiments confirm the hypothesis that pre-selection methods are as important as the social choice algorithm in this scope.

6.3. Experimental results on a case with anomalous preferences

As explained above, random preferences can cause uniform results whatever is decided with whatever method used since, for a configuration chosen, there are always users who find it satisfactory. In this section, the experiments are conducted over an scenario where half the users have random preferences, and half of them mark the configuration "sport" with the highest score (10) and the remaining configuration with the lowest (0). This scenario aims at modeling the broadcasting of an important soccer match in the modeled hotel. In this scenario, there are a number of users whose satisfaction is binary.

The results considering a pre-selection method based on the Euclidean distance are shown in Fig. 14. The basis case, first in arrival, does not include pre-selection to study the benefit obtained by combining voting system plus pre-selection. These results show that the base case is considerably improved in the three metrics considered which any voting system: *as* is increased from 62% to 86% with range voting, *uwc* rises from 1263.18 users to 1914.02 users with range voting, and *mwc* is reduced from 244.55 time steps to 41.4 time steps with cumulative voting. Again, the novel voting method proposed is the second best option in both *as* and *mwc* metrics. The pre-selection methods work poorly in the voting systems whose vote format offers little information about users' preferences such as the plurality and approval methods. Nonetheless, these two voting systems, which have usability remarkably better than range voting and exchange of weights, obtain results reasonable good and considerably better than the base case.

In addition to the VoteSim tool [8], and the experiments results extended [2], there is a video available online [7] showing the use of VoteSim to model this case study and to conduct the experiments presented.

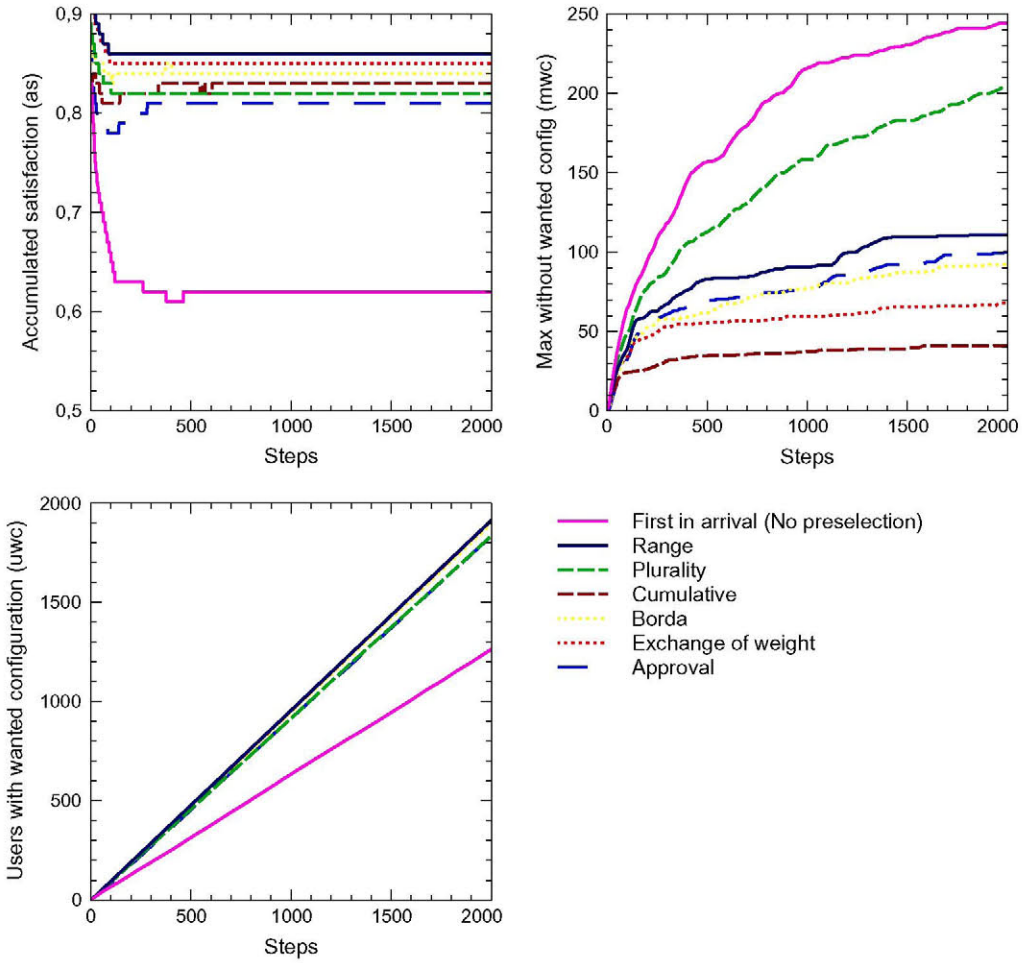

Fig. 14. Experimental results in a population with anomalous preferences, a pre-selection method based on the Euclidean distance, and different voting systems. Tables with experiments results extended are available on-line [2].

7. Conclusions and future work

This paper concludes, by sound and reproducible results, that the use of social choice theory in *Ambient Intelligence* (AmI) systems can improve considerably users' satisfaction when accessing shared resources. The research has been conducted by *agent based social simulations* (ABSS) and presents the following conclusions:

- There are significant differences between the AmI case and the political elections. For example, establishing a comprehensive list of candidates (configurations in the AmI case) is not always possible. Besides, services owners are likely to be interested in avoiding long waits to get a candidate elected when at least a voter wants this candidate. Another difference is that voting is more frequent and the vote is not necessarily secret for the agent in charge of the agreement service. Therefore, *reputation* research is relevant when tactical voters are considered. Finally, since there can be several services of the same type in the environment, the voter can choose the group to vote in. Therefore, *pre-selection* mechanisms can be provided. Moreover, this makes random preferences not cause uniform results regardless the voting systems unlike the political election case.
- As concluded above, different metrics are necessary in this scope. Three of them have been proposed and formalized to evaluate different voting systems in the AmI case: accumulated satisfaction (*as*), number of users with wanted configurations (*uwc*), and the maximum time without a wanted configuration (*mwc*).
- Regarding the performance of well known electoral systems in the AmI case, the results show that the use of range voting achieves 12% more *as* than the basis method which consists of allowing the first user in accessing the resource to decide the configuration. The approval voting, which presents a better usability than range voting, gets 8% more *as*. However, if the service owner is more interested in avoiding long waits, cumulative voting gives the best *mwc*: 98.1 time steps. Although this result is less than half the time required with the basis method, the worst result according to this metric is given by the most commonly used voting method, the plurality method (304.2 t.s.).

- The novel voting algorithm called *exchange of weights* is the most balanced voting system considered. This method not only gets better *mwc* than the cumulative voting, but also gets 5% more *as* than this method. In other scenarios where the use of pre-selection mechanisms and other preference models are considered, the exchange of weight is the second best option for *as* and *mwc*, while a number of voting methods offer similar results in *uwc*.
- The use of greedy pre-selection mechanisms based on Euclidean or Manhattan distances outperform the results given by more complex cluster analysis techniques such as EM and K-means according to the *as* and *uwc* metrics (with their default parameters except the number of clusters). Nonetheless, these latter algorithms improve the former ones in terms of *mwc*. The results show that preselection mechanisms are as important as the voting system employed: the *as* that range voting presents without pre-selection, 69%, reaches 79% with the use of the Euclidean distance; and the *mwc* that cumulative voting achieves drops from 98.1 t.s. to 27.7 t.s.
- When anomalous preferences happen instead of having uniform random preferences, the use of voting systems combined with pre-selection mechanism are even more important: for example, *as* is increased from 62% to 86% with range voting.

To ensure the reproducibility of the experimental results given in this paper, the free and open-source tool *VoteSim* [8] has been presented. Furthermore, the experiments results [2] and a video of the case study [7] are available on-line.

Concerning the future works, most of them have been introduced throughout the paper. There are a number of considerations which would improve the model implemented in VoteSim and which could affect the experiments results. Firstly, the inclusion of location and need theories in the user agent model as explained in Section 4.1.2. Secondly, the consideration of tactical voting and the effect that different populations with different strategies can cause in different voting systems, see Section 4.1.3. Thirdly, there are a large number of voting systems which could be considered besides those introduced in Section 4.2.1. Fourthly, there are also numerous techniques for cluster analysis and group recommender systems [27] that could be integrated into VoteSim in addition to those explained in Section 4.2.4. Fifthly, the inclusion of complex preferences which requires a number of decisions in voting methods as seen in Section 4.2.2. Finally, VoteSim could include displays summarizing executions results [51] to help developers to decide the best voting technique for a specific case.

Despite these future works, this paper hints at the potential of considering social choice techniques in intelligent environments and present the agent based social simulation as a fundamental tool to evaluate the performance of these techniques.

Acknowledgments

This research work is supported by the Spanish Ministry of Economy and Competitiveness under the R&D Projects CALISTA (TEC2012-32457) and Thofu (CEN-2010-1019).

References

[1] CALISTA (Agents Technology and Services Engineering for diagnostics and configuration of home network using a mobile phone). <http://goo.gl/bp1i5J>. Reference: TEC2012-32457.
[2] Experimental results for the case study presented in this paper. <http://www.gsi.dit.upm.es/eserrano/VoteSimExperiments/>.
[3] Glossary of statistical terms, reproducibility. <http://www.statistics.com/index.php?page=glossary&term_id=833>.
[4] NetLogo website. <http://ccl.northwestern.edu/netlogo/>.
[5] THOFU (Tecnologfas para el HOtel del FUturo). <http://www.thofu.es/>. Reference: CEN-2010-1019.
[6] UbikSim website. <https://github.com/emilioserra/UbikSim/wiki>.
[7] VoteSim video. <http://www.youtube.com/watch?v=T5E-hOpxCs>.
[8] VoteSim website. <https://github.com/gsi-upm/VoteSim/wiki>.
[9] J. Alcantud, R. de AndrTs Calle, J. Cascön, On measures of cohesiveness under dichotomous opinions: some characterizations of approval consensus measures, Inform. Sci. 240 (0) (2013) 45–55.
[10] K.J. Arrow, A.K. Sen, K. Suzumura (Eds.), Handbook of Social Choice and Welfare, first ed., vol. 2, Elsevier, 2011.
[11] A.M. Aseere, E.H. Gerding, D.E. Millard, A voting-based agent system for course selection in e-learning, Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology – WI-IAT '10, vol. 02, IEEE Computer Society, Washington, DC, USA, 2010, pp. 303–310.
[12] M. Balinski, R. Laraki, Majority Judgment: Measuring, Ranking, and Electing, Mit Press, 2010.
[13] J. Barton, V. Vijayaraghavan, Ubiwise: A Ubiquitous Wireless Infrastructure Simulation Environment, HP Labs, 2002.
[14] M. Benyoucef, R.K. Keller, An evaluation of formalisms for negotiations in e-commerce, in: Proceedings of the Third International Workshop on Distributed Communities on the Web, DCW '00, Springer-Verlag, London, UK, 2000, pp. 45–54.
[15] R.R. Bouckaert, E. Frank, M. Hall, R. Kirkby, P. Reutemann, A. Seewald, D. Scuse, Weka manual (3.7.1), June 2009. <http://prdownloads.sourceforge.net/weka/WekaManual-3-7-1.pdf?download>.
[16] J. Breuker, W. Van de Velde, CommonKADS library for expertise modelling: reusable problem solving components, in: Frontiers in Artificial Intelligence and Applications, IOS Press, 1994.
[17] F. Buccafurri, V.D. Foti, G. Lax, A. Nocera, D. Ursino, Bridge analysis in a social internetworking scenario, Inform. Sci. 224 (0) (2013) 1–18.
[18] A. Chavez, P. Maes, Kasbah: An Agent Marketplace for Buying and Selling Goods Overview of Kasbah, Media.
[19] D.J. Cook, J.C. Augusto, V.R. Jakkula, Ambient intelligence: technologies, applications, and opportunities, Pervasive Mobile Comput. 5 (4) (2009) 277–298.
[20] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the em algorithm, J. Roy. Stat. Soc. Ser. B 39 (1) (1977) 1–38.
[21] A. Drogoul, D. Vanbergue, T. Meurisse, Multi-agent based simulation: where are the agents?, in: Proceedings of the 3rd International Conference on Multi-agent-based Simulation II, MABS'02, Springer-Verlag, Berlin, Heidelberg, 2003, pp 1–15.
[22] J.J. Durán, C.A. Iglesias, An Ontology for Formalising Agreement Patterns in Auction Markets, 2010.
[23] A. Fip, FIPA ACL message structure specification (SC00061G), FIPA TC Commun. (2002).
[24] P.A. Fishwick, Simulation model design and execution: building digital worlds, in: Prentice Hall International Series in Industrial and Systems Engineering, Prentice Hall, Englewood Cliffs, NJ, 1995.

[25] P.A. Fishwick, Computer simulation: growth through extension, Trans. Soc. Comput. Simul. Int. 14 (1997) 13–23.
[26] J. Garcfa-Lapresta, M. Martfnez-Panero, L. Meneses, Defining the borda count in a linguistic decision making context, Inform. Sci. 179 (14) (2009) 2309–2316.
[27] I. Garcia, S. Pajares, L. Sebastia, E. Onaindia, Preference elicitation techniques for group recommender systems, Inform. Sci. 189 (0) (2012) 155–175.
[28] T. Garcia-Valverde, F. Campuzano, E. Serrano, A. Villa, J.A. Botia, Simulation of human behaviours for the validation of ambient intelligence services: a methodological approach, J. Ambient Intell. Smart Environ. 4 (3) (2012) 163–181.
[29] T. Garcia-Valverde, E. Serrano, J. Botia, Combining the real world with simulations for a robust testing of ambient intelligence services, Artif. Intell. Rev. (2012) 1–24.
[30] N. Gilbert, K.G. Troitzsch, Simulation for the Social Scientist, Open University Press, 2005.
[31] J. Holubiec, G. Szkatula, D. Wagner, A knowledge-based model of parliamentary election, Inform. Sci. 202 (2012) 24–40.
[32] C.A. Iglesias, M. Garijo, J.I. Fernández-Villamor, J.J. Durán, Agreement Patterns, 2009.
[33] T. Ito, H. Hattori, M. Zhang, T. Matsuo, Rational, robust, and secure negotiations in multi-agent systems, in: Studies in Computational Intelligence, Springer, 2008.
[34] N.R. Jennings, Agreement technologies, in: Proceedings of the 33rd Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM '07, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 111–113.
[35] O. Kwon, J.M. Shin, S.W. Kim, Context-aware multi-agent approach to pervasive negotiation support systems, Expert Syst. Appl. 31 (2) (2006) 275–285.
[36] J.-S. Lee, S. Olafsson, A meta-learning approach for determining the number of clusters with consideration of nearest neighbors, Inform. Sci. 232 (0) (2013) 208–224.
[37] O. Linda, M. Manic, Interval type-2 fuzzy voter design for fault tolerant systems, Inform. Sci. 181 (14) (2011) 2933–2950.
[38] H. Liu, H. Darabi, P. Banerjee, J. Liu, Survey of wireless indoor positioning techniques and systems, IEEE Trans. Syst. Man Cybernet. Part C 37 (6) (2007) 1067–1080.
[39] M.S. Mahmoud, H.M. Khalid, Expectation maximization approach to data-based fault diagnostics, Inform. Sci. 235 (0) (2013) 80–96.
[40] E. Mangina, J. Carbo, J. Molina, Agent-based ubiquitous computing, in: Atlantis Ambient and Pervasive Intelligence, We Publish Books, 2010.
[41] W.D. Mulder, Optimal clustering in the context of overlapping cluster analysis, Inform. Sci. 223 (0) (2013) 56–74.
[42] H. Nishikawa, S. Yamamoto, M. Tamai, K. Nishigaki, T. Kitani, N. Shibata, K. Yasumoto, M. Ito, UbiREAL: Realistic smartspace simulator for systematic testing, UbiComp 2006: Ubiquitous Computing, 2006, pp. 459–476.
[43] H.S. Nwana, Software agents: an overview, Knowl. Eng. Rev. 11 (1996) 205–244.
[44] A. Oluyomi, Patterns and protocols for agent oriented software development, PhD, The University of Melbourne, 2006.
[45] E. O'Neill, M. Klepal, D. Lewis, T. O'Donnell, D. O'Sullivan, D. Pesch, A testbed for evaluating human interaction with ubiquitous computing environments, in: Proceedings of the First International Conference on Testbeds and Research Infrastructures for the DEvelopment of NeTworks and COMmunities, TRIDENTCOM '05, IEEE Computer Society, Washington, DC, USA, 2005, pp. 60–69.
[46] A. Paschke, C. Kiss, S. Al-Hunaty, A pattern language for decentralized coordination and negotiation protocols, in: Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'05) on e-Technology, e-Commerce and e-Service, EEE '05, IEEE Computer Society, Washington, DC, USA, 2005, pp. 404–407.
[47] A.D. Procaccia, How is voting theory really useful in multiagent systems? <http://www.cs.cmu.edu/arielpro/papers/vote4mas.pdf>.
[48] L.A. San Martín, V.M. Peláez, R. González, A. Campos, V. Lobato, Environmental user-preference learning for smart homes: an autonomous approach, J. Ambient Intell. Smart Environ. 2 (3) (2010) 327–342.
[49] D.C. Schmidt, Using design patterns to develop reusable object-oriented communication software, Commun. ACM 38 (10) (1995) 65–74.
[50] E. Serrano, J. Botia, Validating ambient intelligence based ubiquitous computing systems by means of artificial societies, Inform. Sci. 222 (0) (2013) 3–24.
[51] E. Serrano, A. Muñoz, J. Botia, An approach to debug interactions in multi-agent system software tests, Inform. Sci. 205 (0) (2012) 38–57.
[52] E. Serrano, A. Quirin, J. Botia, O. Cordön, Debugging complex software systems by means of pathfinder networks, Inform. Sci. 180 (5) (2010) 561–583.
[53] E. Serrano, M. Rovatsos, J.A. Botfa, Data mining agent conversations: a qualitative approach to multiagent systems analysis, Inform. Sci. 230 (0) (2013) 132–146.
[54] E. Serrano, M. Rovatsos, J. Botia, A qualitative reputation system for multiagent systems with protocol-based communication, Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems – AAMAS '12, vol. 1, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2012, pp. 307–314.
[55] E. Serrano, J.M. Such, J. Botía, A. García-Fornes, Strategies for avoiding preference profiling in agent-based e-commerce environments, Appl. Intell. 40 (1) (2014) 127–142.
[56] Y. Shoham, K. Leyton-Brown, Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations, Cambridge University Press, New York, NY, USA, 2008.
[57] R.G. Smith, The contract net protocol: high-level communication and control in a distributed problem solver, IEEE Transactions on Computers, vol. C-29, IEEE Computer Society, Washington, DC, USA, 1980, pp. 1104–1113.
[58] S.Y.W. Su, C. Huang, J. Hammer, A replicable web-based negotiation server for e-commerce, in: Proceedings of the 33rd Hawaii International Conference on System Sciences – vol. 8, HICSS '00, IEEE Computer Society, Washington, DC, USA, 2000, p. 8016.
[59] V.A.M. Tamma, S. Phelps, I. Dickinson, M. Wooldridge, Ontologies for supporting negotiation in e-commerce, Eng. Appl. AI 18 (2) (2005) 223–236.
[60] L. Tang, X. Zhou, C. Becker, Z. Yu, G. Schiele, Situation-based design: a rapid approach for pervasive application development, in: UIC/ATC, 2012, pp. 128–135.
[61] D.I. Tapia, J.A. Fraile, S. Rodrfguez, R.S. Alonso, J.M. Corchado, Integrating hardware agents into an enhanced multi-agent architecture for ambient intelligence systems, Inform. Sci. 222 (0) (2013) 47–65.
[62] M. Woolridge, M.J. Wooldridge, Introduction to Multiagent Systems, John Wiley & Sons, Inc., New York, NY, USA, 2001.