VirtualECare: Group Support in Collaborative Networks Organizations for Digital Homecare

Ricardo Costa[1], Paulo Novais[2], Luís Lima[1], José Bulas Cruz[3] and José Neves[2]

[1]College of Management and Technology - Polytechnic of Porto, Felgueiras, Portugal

2DI-CCTC, Universidade do Minho, Braga, Portugal

[3]University of Trás-os-Montes e Alto Douro, Vila Real, Portugal

Abstract

Collaborative Work plays an important role in today's organizations and normally in areas where decisions must be made. However, any decision that involves a collective or group of decision makers is, by itself, complex but is becoming normal in recent years. In this work we present the *VirtualECare* project (Figure 10), intelligent multi-agent system able to monitor, interact and serve its customers, which are, normally, in need of care services.

In the last years there has been a substantially increase in the number of people needed of intensive care, especially among the elderly, a phenomenon that is related to population ageing. However, this is not exclusive of the elderly, as diseases as obesity, diabetes, and blood pressure have been increasing among young adults. As a new reality, it has to be dealt by the health sector, and particularly by the public one. Thus, the importance of finding new and cost effective ways for health care delivery are of particular importance, especially when one want them not to be removed from their "habitat". Following this line of thinking, the *VirtualECare* project will be presented, like similar ones that preceded it.

Recently we have assisted to a growing interest in combining the advances in information society - computing, telecommunications and presentation – in order to create Group Decision Support Systems (GDSS). Indeed, the new economy, along with increased competition in today's complex business environments, takes the companies to seek complementarities in order to increase competitiveness and reduce risks. Under these scenarios, planning takes a major role in a company life. However, effective planning depends on the generation and analysis of ideas (innovative or not) and, as a result, the idea generation and management processes are crucial.

Our objective is to apply the above presented GDSS to a new area. We believe that the use of GDSS in the healthcare arena will allow professionals to achieve better results in the analysis of one's Electronically Clinical Profile (ECP). This achievement is vital, regarding the explosion of knowledge and skills, together with the need to use limited resources and get better results.

Costa R., Novais P., Lima L., Bulas-Cruz J. and Neves J., VirtualECare: Group Support in Collaborative Networks Organizations for Digital Homecare, in Handbook of Digital Homecare, Springer Series: Series in Biomedical Engineering, Yogesan, K.; Bos, L.; Brett, P.; Gibbons, M.C. (Eds.), ISBN: 978-3-642-01386-7, pp 151-178, 2009.

Table of Contents

Introduction

Once the human population is ageing, it is a matter of fact that the elderly in need of special attention is growing. Old age brings new problems (e.g., health, loneliness), aggravated with the lack of specialized human resources to assist their needs. However, this is not exclusive of the elderly, as diseases like obesity, diabetes, and blood pressure have been increasing among young adults [1]. As a new reality, it has to be dealt with by the health sector, and especially by the public one. Thus, the importance of finding new and cost effective ways for health care delivery are of particular importance, especially when one wants them not to be removed from their "habitat" [2]. Besides that fact, pressures exist in government and society (e.g., budgetary restraints, cost of medical technologies and cost of internment) that will force readjustments of actual health care practice, which may also affect other co-related public systems [3, 4].

Motivation

In the last years we have assisted to a proliferation of various research projects in order to increase the quality of care services and reduce the associated costs, especially the ones that require the patient to be delocalized from his natural habitat (Home). Normally these tend to be simple and basic reactive alarm systems without many requirements from the support platform point of view [5]. In our opinion these systems were very useful to delineate a path for others to follow. Taking this path we have presented the VirtualECare project [6, 7] which we believe will be the next generation of remote proactive healthcare system with, in our case, Group Decision techniques for problem solving through the use of today's available, low cost, technology making this way a very promising approach to a possible solution for some of the health sector problems.

Collaborative Networks in Digital Homecare

The use of collaborative networks in the care of the elderly may be an important part of a social development process, yet it has not been studied in depth. This work looks at the role that collaborative networks and learning plays within the innovative processes of a smart home for care of the elderly, and suggests a framework that will allow an organization to strategically model a collaborative environment that may be conducive to innovation. Such a framework will identify key areas of the Inter-Virtual Organizations Co-operation for Care of the Elderly, which should be discussed in line with the collaborative tool requirements of the care providers. A theoretical ontology based tool is also briefly discussed to capture and identify how the services of the elderly project team are innovating and provide care providers with collaborative tools, which will reflect their collaborative and knowledge needs.

Some work on the above problem has been made, namely using alarm systems that can be triggered by the monitored people in case of necessity, to more modern ones, using almost any artifacts that the new technologies have to offer [4, 8].

The major goal in our work is to take the work already done to a next level, enhancing elderly quality of life [3]. The path to pursue relies on a mix of different contributions from Artificial

Intelligence, such as Collaborative Networks, Ambient Intelligence and Knowledge Representation tools coupled whit different computational paradigms and methodologies for problem solving, such as Agent Based Systems and Group Decision Support Systems. To achieve such a result, we will enrich any space (e.g., houses, buildings, critical areas in hospitals) with smart artifacts so that through the use of automated or semi-automated Group Decision Support Systems we may diagnose healthcare problems (and more) and present solutions on time [9].

The challenges faced by both business and academy in recent years, in association with the advances in information and communication technology, lead to the creation of a large variety of Networks, namely Collaborative Networks (CN). Basically, CN let professionals and organizations to seek complementary and joint activities, allowing them to participate in new and more competitive businesses opportunities, reaching new markets and/or fostering scientific excellence, either in forms of services or products. This can be done, namely, through highly integrated supply chains, virtual enterprises/organizations, professional virtual communities, value constellations and/or virtual laboratories [10].

Group Decision Support Systems

Group Support Systems

By definition, any Collaborative Network Organizations (CNO) has to support collaborative work, that presupposes the existence of a group of people that has as mission the completion of a specific task [11]. The number of elements involved in the group may be variable, as well as the persistency of the group. The group members may be at different places, meet in an asynchronous way or may belong to different organizations. Collaborative work has not only inherent advantages (e.g., greater pool of knowledge, different world perspectives, increased acceptance), but also assertive goals (e.g., social pressure, domination, goal displacement, group thinking) [12].

Meeting phases

Group Support Systems (GSS) intend, as we shall see, to support collaborative work. In this work we will call “meeting” to all the processes necessary to the completion of a specific collaborative task. A meeting is a consequence or an objective of the interaction between two or more persons [13]. Physically, a meeting can be realized in one of the four scenarios: same time / same place, same time / different places, different times / same place and different times / different places. Each one of these scenarios will require from the GSS a different kind of support.

Until now we discussed collaborative work and present group members as the only persons involved in the process. However, it is very common to see a third element taking part in the course of action, the facilitator. The meeting facilitator is a person welcomed by all the members of the group, neutral and without authority to make decisions, which intervenes in the process in order to support the group in the identification of a problem and in the finding of a solution, in order to increase group efficiency [14].

According to Dubs and Hayne [15] a meeting has three distinct phases, as it is depicted in Figure 1.

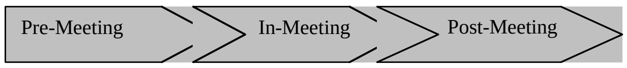


Figure 1. Meeting Phases

In the Pre-Meeting phase the facilitator prepares the meeting, i.e., establishes the meeting goals, proceeds with the group formation (making sure that all the participants have the necessary background), selects the best supporting tools, informs the meeting members about the goals and distributes among them the meeting materials.

In the In-Meeting phase the participants will be working in order to accomplish the meeting goals, and the facilitator has the task of monitoring the meeting interactions (e.g., to observe the relationship between the group members) and to intervene if necessary.

In the Post-Meeting phase, it is important to evaluate the results achieved by the group, as well as by how much each group member is acquit with the achieved results (satisfied/unsatisfied). Still, in this phase it is very important to identify and store information that can be useful in future meetings (e.g., how to actualize the participant's profile for future selection).

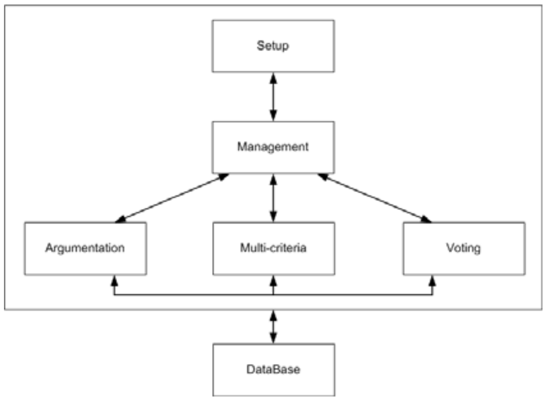
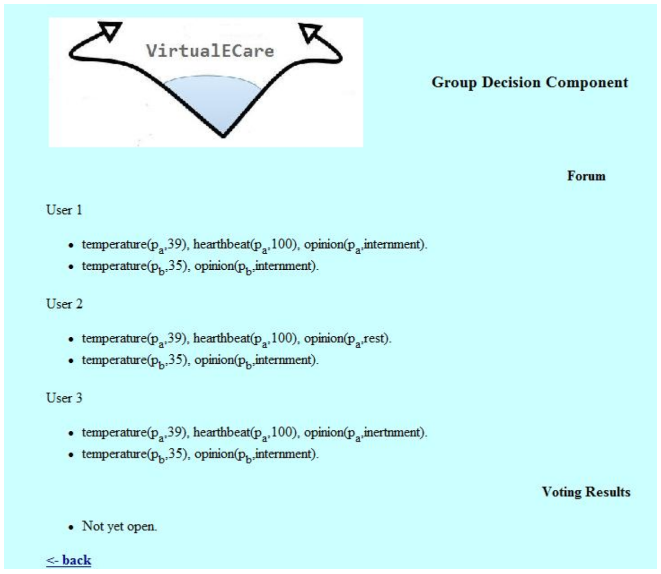


Figure 2. VirtualECare Group Decision Architecture

Setup module – will be operated by a *facilitator* during the pre-meeting phase that will do several configuration and parameterization activities;

Multi-criteria module – will be operated by a *facilitator* during the pre-meeting phase, being in charge of the definition of the evaluation criteria and scales and, eventually, in deleting dominated alternatives.

Argumentation module - This module is based on the IBIS (Issue Based Information System) argumentation model developed by [16] and his colleagues in the early 70's. According to this model, an argument is a statement or an opinion which may support or pointed out one or more ideas (Figure 3).



Forum

User 1

- temperature(p_a,39), hearthbeat(p_a,100), opinion(p_a,internment).
- temperature(p_b,35), opinion(p_b,internment).

User 2

- temperature(p_a,39), hearthbeat(p_a,100), opinion(p_a,rest).
- temperature(p_b,35), opinion(p_b,internment).

User 3

- temperature(p_a,39), hearthbeat(p_a,100), opinion(p_a,inertnment).
- temperature(p_b,35), opinion(p_b,internment).

Voting Results

- Not yet open.

<- back

Figure 3. Forum

Voting module - This module is responsible for allowing each intervenient of the decision group component to "vote" for his preferred choice, normally the one most similar to his "opinion" (Figure 3).

Recommendation System

Idea Generation

The ***Group Decision*** module, as it was said before, is a major module of our system. This fact, associated with the importance of decision-making in today business activity and with the needed celerity in obtaining a decision in the majority of the cases that this key module will be defied to resolve, requires a real effectiveness of the decision making process. Thus, the need for an Idea Generation tool that will support the meetings, being those face to face, asynchronous or distributed, becomes crucial.

The flow of new ideas is central in an environment as the one presented above. Several idea generation techniques were popularized during the early 1950's in order to assist organizations to be fully innovative. These techniques, although primarily born and used in the advertising world, can be applied to an infinite number of emerging areas. Many idea techniques emerged from that time and continue to current days, such as Brainstorming, Nominal Group Technique (NGT), Mind-mapping and SCAMPLER, among others.

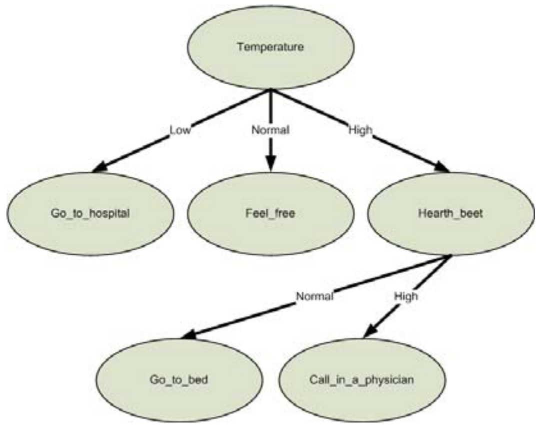


Figure 4. A decision tree of a specific problem

In order to face the real challenges that this module have to deal with, we selected two idea generation techniques for different situations:

Brainstorming – it is probably the best-known creative tool. It can be used in most groups, although in most cases the rules that oversee it must be perceived by the group elements. It comes with all its potential when and independent facilitator manages the process (so the group can focus on the creative tasks). Normally, a brainstorming has duration somewhere between 30 minutes to 1 hour, depending on the difficulty of the problem and the motivation of the decision group. Due to this fact it cannot be used in situations of life or death, but it can and is going to be used in assessing patient's quality of life;

Mind-mapping – it is best used when one needs to explore and/or develop ideas for a specific problem, or when we need to take notes and/or summarize meetings. It can be used to obtain immediate answers in critical situations.

In Mind-mapping the specific problem is presented in the form of a decision tree, being the vital data obtained, for instance, from the sensors attached to the **Supported User** (Figure 4).

Argumentation

After establishing individual ideas (through the above presented tools, or simply by intuition) the participants are expected to "defend" those ideas in order to reach consensus or majority. Each participant will, therefore, and in a natural way, argue for the most interesting alternatives or against the worst alternatives, according to his/her preferences and/or skills. By expressing their arguments, participants expect to influence the others' opinions and make them change their own [17].

This module is based on the IBIS (Issue Based Information System) argumentation model developed by Rittel and his colleagues in the early 70's [18]. The core of this methodology is based on the matrix of questions, ideas and arguments that, all combined, represent a dialogue. According to this model, an argument is a statement or an opinion which may support or pointed out one or more ideas.

Among the three elements of the IBIS model, there exists nine possible links, as it is depicted in Figure 5

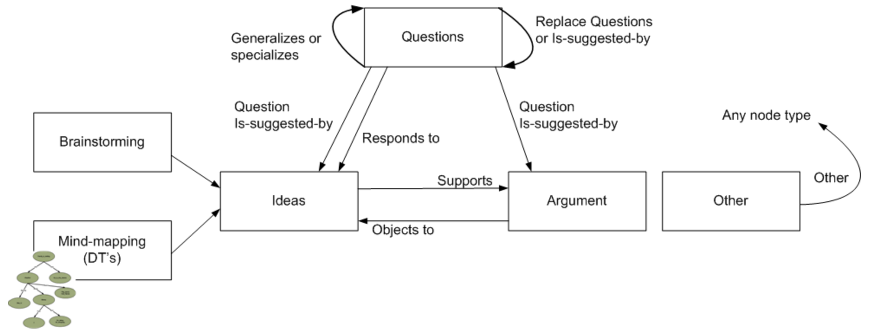


Figure 5. IBIS model adapted from Conklin and Begeman

In the implementation process of the Group Decision apparatus, and the respective support software, some modifications to the model have been made.

The **question** in the IBIS model is, in the Group Decision apparatus, the **goal** of the meeting.

Ideas are the alternatives of the multi-criteria decision problem and arise from the idea generation tool throughout brainstorming or through mind mapping.

Arguments in IBIS can be pros or cons vis-à-vis a given idea. In the Group Decision module they are based in two types of information: Patient Electronic Clinical Profile and a set of Decision Trees. Additionally, the possibility for one participant to argue using an argument from another member is real.

This module is paramount on the in-meeting phase. It is used by the participants to defend their positions, but can also be used in the post-meeting phase by the facilitator (e.g. if the group does not reach a solution, the facilitator may use this module to check which is the most consensual alternative).

The IBIS model has been often used in the development of GDSSs, the first implementation being gIBIS [19]. By adopting this model, the Group Decision module should enable a better organization of the arguments exchanged by the participants. This may facilitate opinion convergence, and at the same time to reduce the meetings "noise".

Once a decision has been made, it is (automatically) sent to the monitored person (***Supported User*** in Figure 10) by a mobile device (Figure 6), in order to keep him/her informed.

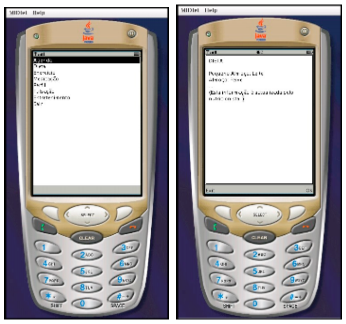


Figure 6. A Group Decision assessment reported on a person mobile device

Quality of Information

VirtualECare Group Decision Support System deals with information and knowledge in an environment of uncertainty. The information available must always be considered incomplete and imperfect.

How does a decision maker is confident about the reliability of the information at hand? In group decisions each person that participates in the final decision must be confident on: The reliability of the computer support system; The other decision makers; The information rolling in and out of the system and the information exchanged between participants. The Group Decision of the VirtualECare system operates in such environment. We leave the first issue to others and concentrate in the last two, proposing a model for computing the quality of information.

A suitable representation of incomplete information and uncertainty is needed, one that supports non-monotonic reasoning. In a classical logical theory, the proof of a theorem results in a *true* or *false* truth value, or is made in terms of representing something, with respect to one may not be conclusive. In opposition, in a logic program, the answer to a question is only of two types: *true* or *false*. This is a consequence of the limitations of the knowledge representation in a logic program, because it is not allowed explicit representation of negative information. Additionally, the operational semantics applies the Closed-World Assumption (CWA) [20, 21] to all the predicates. The generality of logic programs represents implicitly negative information, assuming the application of reasoning according to the CWA.

An extended logic program, on the other hand, is a finite collection of rules of the form [22, 23]:

$q \leftarrow p_1 \wedge \ldots \wedge p_m \wedge not\ p_{m+1} \wedge \ldots \wedge not\ p_{m+n}$	(1)
$?\, p_1 \wedge \ldots \wedge p_m \wedge not\ p_{m+1} \wedge \ldots \wedge not\ p_{m+n}$	(2)

where ? is a domain atom denoting falsity, the pi, qj, and p are classical ground literals, i.e. either positive atoms or atoms preceded by the classical negation sign ¬. Every program is associated with a set of abducibles. Abducibles can be seen as hypotheses that provide possible solutions or explanations of given queries, being given here in the form of exceptions to the extensions of the predicates that make the program.

The objective is to provide expressive power for representing explicitly negative information, as well as directly describe the CWA for some predicates, also known as *predicate circumscription* [24]. Three types of answers to a given question are then possible: *true*, *false* and *unknown* [21]. The representation of null values will be scoped by the ELP. In this work, we will consider two types of null values: the first will allow for the representation of unknown values, not necessarily from a given set of values, and the second will represent unknown values from a given set of possible values. We will show now how null values can be used to represent unknown information. In the following, we consider the extensions of the predicates that represent some of the properties of the participants, as a measure of their skills for the decision making process:

```
area of expertise: Entities x StrValue
role: Entities x StrValue
credible: Entities x Value
reputed: Entities x Value
```

The first argument denotes the participant and the second represents the value of the property (e.g., `credible(luis, 100)` means that the credibility of the participant `luis` has the value `100`).

```
credible(luis,100)
¬credible(E,V)←
             not credible(E,V)
```

Program 1. Extension of the predicate that states the credibility of a participant

In Program 1, the symbol ¬ represents the strong negation, denoting what should be interpreted as false, and the term *not* designates negation-by-failure.

Let us now admit that the credibility of another possible participant *ricardo* has not, yet, been established. This will be denoted by a null value, of the type unknown, and represents the situation in Program 2: the participant is credible but it is not possible to be certain (affirmative) about its value. In the second clause of Program 2, the symbol ⊥ represents a null value of an undefined type. It is a representation that assumes any value as a viable solution, but without being given a clue to conclude about which value one is speaking about. It is not possible to compute, from the positive information, the value of the credibility of the participant *ricardo*. The fourth clause of Program 2 (the closure of predicate credibility) discards the possibility of being assumed as false any question on the specific value of credibility for participant *ricardo*.

```
credible(luis,100)
credible(ricardo,⊥)
¬credible(E,V)←
             not credible(E,V),
             not exception(credible(E,V))
exception(credible(E,V))←
             credible(E,⊥)
```

Program 2. Credibility about participant ricardo, with an unknown value

Let's now consider the case in which the value of the credibility of a participant is foreseen to be 60, with a margin of mistake of 15. It is not possible to be positive, concerning the credibility value. However, it is false that the participant has a credibility value of 80 or 100. This example suggests that the lack of knowledge may only be associated to a enumerated set of possible known values. As a different case, let's consider the credibility of the participant *paulo*, that is unknown, but one knows that it is specifically 30 or 50.

```
credible(luis,100)
credible(ricardo,⊥)
¬credible(E,V)←
            not credible(E,V),
            not exception(credible(E,V))
exception(credible(E,V))← credible(E,⊥)
exception(credible(carlos,V))← V ≥ 45 ∧ V ≤ 75
exception(credible(paulo,30))
exception(credible(paulo,50))
```

Program 3. Representation of the credibility of the participants carlos and paulo

Using Extended Logic Programming, as the logic programming language, a procedure given in terms of the extension of a predicate called *demo* is presented here. This predicate allows one to reason about the body of knowledge presented in a particular domain, set on the formalism previously referred to. Given a question, it returns a solution based on a set of assumptions. This meta predicate is defined as: `Demo: Question x Answer`

Where Question indicates a theorem to be proved and Answer denotes a truth value (see Program 4): true (T), false (F) or unknown (U).

```
demo(Q,T)← Q
demo(Q,F)← ¬Q
demo(Q,U)← not Q ∧ not ¬Q
```

Program 4. Extension of meta-predicate demo

Let *i* ($i \in 1,\ldots, m$) represent the predicates whose extensions make an extended logic program that models the universe of discourse and *j* ($j \in 1,\ldots, n$) the attributes of those predicates. Let $x_j \in [min_j, max_j]$ be a value for attribute *j*. To each predicate is also associated a scoring function $V_{ij}[min_j, max_j] \rightarrow 0 \ldots 1$, that gives the score predicate *i* assigns to a value of attribute *j* in the range of its acceptable values, i.e., its domain (for simplicity, scores are kept in the interval [0 … 1]), here given in the form: *all(attribute_exception_list, sub_expression, invariants)*

This denotes that *sub_expression* should hold for each combination of the exceptions of the extensions of the predicates that represent the attributes in the *attribute_exception_list* and the *invariants*.

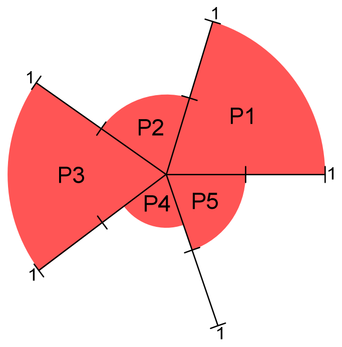


Figure 7. A measure of the quality of information for a logic program or theory P

This is further translated by introducing three new predicates. The first predicate creates a list of all possible exception combinations (pairs, triples, ..., n-tuples) as a list of sets determined by the domain size (and the invariants). The second predicate recurses through this list and makes a call to the third predicate for each exception combination. The third predicate denotes sub_expression, giving for each predicate, as a result, the respective score function. The Quality of Information (QI) with respect to a generic predicate P is therefore given by $QI_P = 1/Card$, where Card denotes the cardinality of the exception set for P, if the exception set is not disjoint. If the exception set is disjoint, the quality of information is given by:

$QI_P = \frac{1}{C_1^{Card} + \cdots + C_{Card}^{Card}}$	(3)

where C_{Card}^{Card} is a card-combination subset, with *Card* elements.

The next element of the model to be considered is the relative importance that a predicate assigns to each of its attributes under observation: w_{ij} stands for the relevance of attribute *j* for predicate *i* (it is also assumed that the weights of all predicates are normalized, i.e.:

$\forall i \sum_{j=1}^{n} w_{ij} = 1$	(4)

It is now possible to define a predicate's scoring function, i.e., for a value $x = (x_1, ..., n)$ in the multi dimensional space defined by the attributes domains, which is given in the form:

$V_i(x) = \sum_{j=1}^{n} w_{ij} * V_{ij}(x_j)$	(5)

It is now possible to measure the QI that occurs as a result of a logic program, by posting the $V_i(x)$ values into a multi-dimensional space and projecting it onto a two dimensional one.

Using this procedure, it is defined a circle, as the one given in Figure 3. Here, the dashed n-slices of the circle (in this example built on the extensions of five predicates, named as p_1 ... p_5) denote de QI that is associated with each of the predicate extensions that make the logic program. It is now possible to return to our case above and evaluate the global credibility of the system. Let us consider the logic program (Program 5).

```
¬credible(E,V)← not credible(E,V),
             not exception(credible(E,V))
exception(credible(E,V))← credible(E,⊥)
credible(luis,100)
credible(ricardo,⊥)
exception(credible(carlos,V))← V ≥ 45 ∧ V ≤ 75
exception(credible(paulo,30))
exception(credible(paulo,50))
role(luis,⊥)
role(ricardo,doctor)
exception(role(carlos,doctor))
exception(reputed(luis,80))
exception(reputed(luis,50))
exception(reputed(ricardo,40))
exception(reputed(ricardo,60))
reputed(carlos,100)
```

Program 5. Example of universe of discourse

As an example we represent the QI associated with participants *luis* and *ricardo*, depicted in Figures 4 and 5.

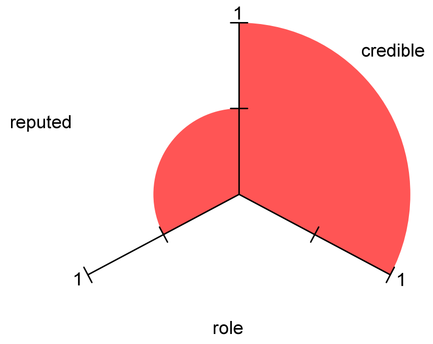


Figure 8. A measure of quality of information about participant luis

In order to find the relationships among the extensions of these predicates, we evaluate the relevance of the QI, given in the form $V_{credible}(luis) = 1$; $V_{reputed}(luis) = 0.785$; $V_{role}(luis) = 0$. It is now possible to measure the QI associated to a logic program referred to above: the shaded n-slices (here n is equal to three) of the circle denote the QI for predicates *credible*, *reputed* and *role*. However, in order to accomplish the main goal of this work, we need to further extend the purpose of Figures 4 and 5, i.e., we may define a new predicate, *trustworthiness*; whose extension may be given in the form of the example (Program 6).

```
¬trustworthiness (X,Y)←
            not trustworthiness (X,Y),
            not exception(trustworthiness (X,Y))
trustworthiness(luis,((credible,1),(reputed,0.785)(role
,0)))
trustworthiness(ricardo,((credible,0),(reputed,0.785),(
role,1)))
```

Program 6. Measuring the global quality

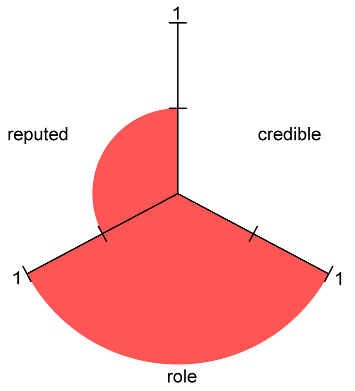


Figure 9. A measure of quality of information about participant ricardo

Besides being able to evaluate the quality of individual actors and individual pieces of information that flows in a group decision system, we aim to have an overall mechanism that allows one to measure the global quality of the system itself and, consequently, the outcomes from it. There is too much in stake when we deal with healthcare, and one must raise the confidence on decisions, especially in an environment of uncertainty, incomplete and imperfect information. The same mechanism used to evaluate individual parts of the system is consistently used to evaluate all the system, through an extension process.

Applications Scenarios

The main goal of VirtualECare is to improve end user's quality of life allowing them to enjoy the so-called active ageing. To achieve this purpose we will take advantage of the enormous evolution new technologies have assisted in past years.

To better understand the amplitude of VirtualECare, let's consider the following scenario [25]:

"John has a heart condition and wears a smart watch that takes his blood pressure three times a day. His watch also reminds him to take his medications and the proper dosage for each medicine. If anything is unusual, his watch alerts both him and the Group Decision Support System (GDSS). John also has a PDA that contains an interactive health control table where he can monitor his medications, schedule his exercises, manage his diet and log his vital statistics. The GDSS has access to this table so they can keep up to date on his condition. Currently, John's watch detects that his blood pressure is unusually high. The GDSS receives a grade B and calls him to check what might be causing his high blood pressure (diagnose). At the same time John receives a checklist of possible causes to review. John compares this list to his own health control table in his PDA to see what might be wrong. Meanwhile, the GDSS decides John should come to an appointment."

The presented scenario requires an infrastructure to support all the several intervenient and provide basic interaction mechanisms. On top of this infrastructure an extensive number of services can be deployed and/or be developed.

VirtualECare Project

Our objective is to present an intelligent multi-agent system not only to monitor and to interact with its costumers (being those elderly people or their relatives), but also to be interconnected to other computing systems running in different healthcare institutions, leisure centres, training facilities or shops. The VirtualECare [9] architecture is a distributed one, being their components unified through a network (e.g., LAN, MAN, WAN), and each one with a different role (Figure 1):

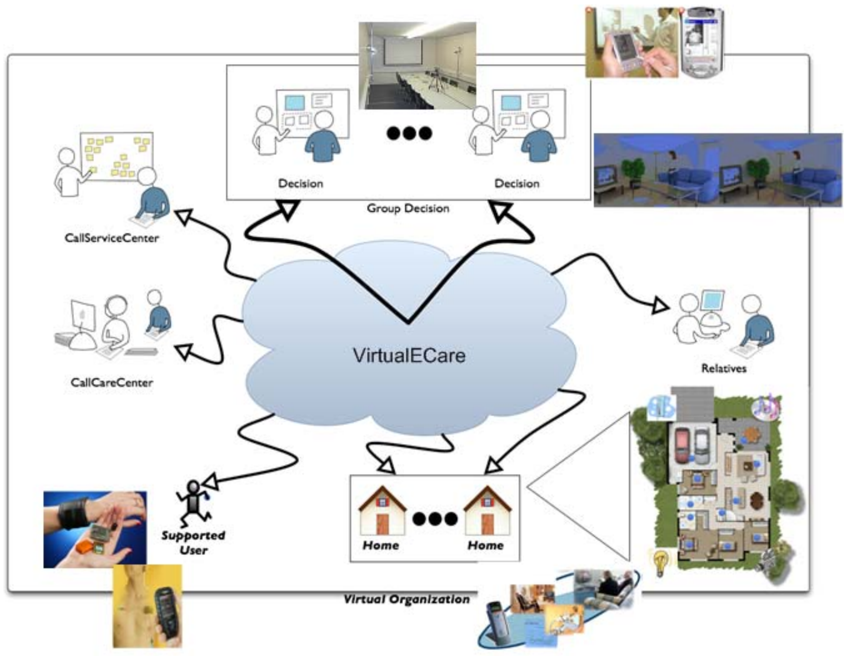


Figure 10. VirtualECare

SupportedUser – Elderly people with special healthcare needs, whose critical data is sent to the ***CallCareCenter*** and forwarded to the ***Group Decision Supported System***;

Home – SupportedUser natural premises. The data collected here is sent to the ***Group Decision Supported System*** through the ***CallCareCenter,*** or to the ***CallServiceCenter (***which speak for themselves);

Group Decision – It is in charge of all the decisions taken at the VirtualECare platform. Our work will be centred on this key module;

CallServiceCenter – Entity with all the necessary computational and qualified personal resources, capable of receiving and analyze the miscellaneous data and take the necessary actions according to it;

CallCareCenter – Entity in charge of the computational and qualified personal resources (i.e., healthcare professionals and auxiliary personnel), capable of receiving and analyze the clinical data, and to take the necessary actions.

Relatives SupportedUser - Relatives which may and should have an active role in the supervising task of their love ones, providing precious complementary information (e.g., loneliness).

In order to the Group Decision Support System take their decisions, one needs of a digital profile of the SupportedUser, which may provide a better understanding of his/her special needs. In this profile we may have different types of data, ranging from the patient Electronic Health Record to their own personal experiences and preferences (e.g., musical, gastronomic). It will provide tools and methodologies for creating an information-on-demand environment that can improve quality-of-living, safety, and quality-of-patient care.

Technology Overview

Infrastructure

Considering the above scenario, and the needs it implies, we have designed a first proposal of a generic, configurable, flexible and scalable infrastructure as presented in Figure 11. It is expectable that on top of it an extensive number of services will progressively arise. These services must, and will be, developed as Web Services, thus allowing the coexistence of several, different, software languages interacting with each other through the use of common messages.

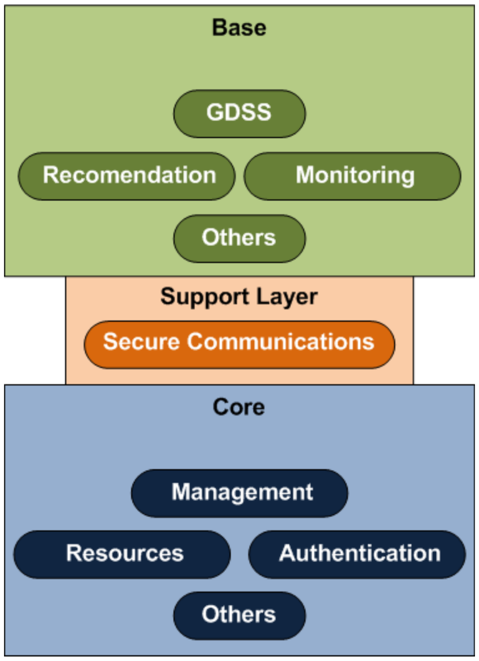


Figure 11. VirtualECare Infrastructure

The fundamental components of the proposed infrastructure are:

- *Secure Communications* – in order to all the components interact, a secure communication infrastructure is mandatory;
- *Management* – responsible for configure and monitor the involved components;

- *Resources* – responsible for every component registration and manage the resources catalog;
- *Authentication* – every component must authenticate itself in order to be able to interact with others;
- *Recommendation* – responsible to make problem solving recommendations;
- *Monitoring* – responsible for interacting with all the sensors and report it's results to the GDSS;
- *GDSS* – responsible for Decision Making.

Architecture

The VirtualECare architecture is a distributed one, composed of a series of different elements eventually geographically separated (Figure 12). It is also a dynamic one since elements can enter and leave at any time, logically or geographically, or the services they provide may vary. The main components of the architecture are the End User and its House, a Monitoring module, the Recommendation System, the Group Decision, the Database, an HL7 module, among others. Each element of the architecture may be very different in its functionalities or software language, calling for an heterogeneous architecture. These are the main issues that were addressed and are detailed in this section: how we make our architecture distributed, modular, dynamic, extensible, flexible, scalable and compatible. To achieve this, we adopted open and widely used technologies and standards, such as OSGi, R-OSGi, FIPA or Web Services.

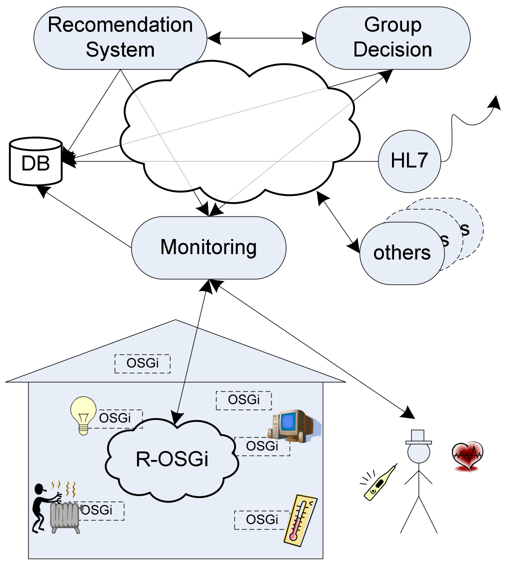


Figure 12. VirtualECare Architecture

To ensure the communication and compatibility between the different components, the Web Services paradigm was used. Web Services can be seen simplistically as a way of sharing information over a network and they are platform independent, being ideal for this kind of systems. Each of the components which provide information declares Web Services that are then requested by the other components which need to access that information. A component can, however, be at the same time a server and a client. The Recommendation System, for example, uses Web Services provided by both the House and the Database and provides, as a service, the Recommendation that is used by the Group Decision. The communication protocol and examples of communication sequences and messages needed for all this components to work together are detailed further ahead.

In Figure 12 we can look at a simplified view of our architecture. The arrows represent Web Services which allow the several components to exchange information. The arrows can be seen as “uses service from” pointing from the client to the server. The House is a little more detailed, showing OSGi and R-OSGi sub-components responsible for interconnecting different kinds of elements

Let us now detail the technologies used in the components by moving to a more close view of the architecture. At this level, two well known standards where used: OSGi and R-OSGi. OSGi is an initiative that intends to establish standards in Java programming, highly specific, catering for the sharing of Java classes, that may be achieved in terms of a services platform paradigm [26, 27]. The use of this technology will let developers build Java applications on a modular basis. The resulting modules are called bundles, which are not only competent to provide services, but also to use services provided from other bundles. In OSGi, a bundle can be installed, started, stopped or un-installed at run-time and without any kind of system reboot, which makes OSGi-based technologies very modular and dynamic.

R-OSGi is an extension to OSGi which allows the access of services provided in remote OSGi implementations, in a completely transparent way, much like they were local services. But what are this technologies good for in our case?

OSGi and R-OSGi are used in our architecture to achieve two main objectives at the level of each component: grant the compatibility and communication between the different parts that make up each component (much like we need to do at a higher level) and establish a logical organization inside the component. These issues come from the multitude of parts that each component can be made of.

Let us look, as an example, to the House component. The House is made of physical parts like 1-Wire sensors and X10 actuators and logical ones like Multi-agent Systems (MAS). 1-Wire is used for measuring environmental values and X10 to control appliances and equipments. MAS are responsible for taking basic reactive actions like control the temperature or call for help in case of need. Moreover, the house may have a big number of rooms and floors which, like the rest of the components, can vary along the time. There is, firstly, the need to organize all these components logically. In order to achieve this we create several OSGi implementations. For each group of similar sensors in each room, a bundle is created. This means that for each room, there will be a bundle reading values from the temperature sensors, another one reading the values from the luminosity sensors and so on. These bundles provide, as a service, the mean value of the last values obtained from the respective sensors. As for the actuators, there is one bundle controlling each equipment or appliance, which is able of sending X10 commands to the equipment it controls. The services these bundle provide are the X10 commands that can be issued to each equipment.

The bundles of the same type in each floor run in the same OSGi implementation, i.e., in each floor there is an OSGi implementation for the temperature bundles, another one for the luminosity bundles, and so on. Likewise, there is an OSGi implementation for the appliances of each type on each floor, i.e., an OSGi for lights, another for air conditioning, etc. In addition, there can also be OSGi implementations inside the House that are just software, like the Multi-agent System that is responsible for taking the basic actions. How we adapted our MAS to be fully integrated with OSGi is described further ahead. Each of these OSGi implementations has at least one additional bundle: a R-OSGi bundle which provides remote access to the implementation. This bundle acts as the bridge between the exterior and the sensors or actuators controlled by the implementation. Its services are the operations that can be performed on the components controlled by the implementation it is in. In the case of the sensors, this bundle is remotely requested to provide the values of sensors and, in the case of the actuators, is through this bundle that the X10 commands arrive to the correct appliance. In this case, the bundle also guarantees that the command is valid and that the consequences of it being executed don’t go against pre-established security policies (e.g. establishing the temperature of the air conditioning in some room to a dangerous value). R-OSGi is therefore the way of integrating each piece inside an OSGi-based component of the architecture, granting the communication between OSGi implementations.

Finally, let’s describe how a MAS is merged inside this system. The MAS is responsible for regularly checking the values of sensors, acting on the actuators accordingly (e.g. the temperature suddenly dropped, turn the heat on) and calling for help in case of need, as well as registering all the events and decisions taken into the Database. Let’s now see how we integrate our MAS with the rest of the architecture. The aim is to make accessible the functionalities of an agent (e.g., its methods) as services to other bundles. It would not be advisable to convert each agent into an OSGi bundle, since it would increase the development time and throw away the advantages of MAS based methodologies for problem solving. Therefore, the decision was to create an OSGI bundle that could make the bridge between regular bundles and Jade: the MAS bundle. This bundle can deal with one Agent Container (AC) and implement the methods declared in the interface of the agents in that AC as its own services. Moreover, this bundle must be able to start and stop agents, which in practice, corresponds to the start and stop of the services provided by them. The bundle, upon the reception of an invocation for an offered service from any other bundle, sends the invocation to the correspondent agent and delivers the respective result to the calling bundle. It must be noted that an agent, when trying to satisfy an invocation, may require the services provided by other bundles currently available. This is possible through the MAS bundle.

As for the interface between the MAS bundle and the Jade system, a JadeGateway agent (JGa) is being used. The task of this agent is to act as a bridge between Jade and non-Jade code. This agent is created when the MAS bundle is started, along with the other agents. The JGa has the knowledge of which services are provided by each agent running so, whenever a request from a service arrives to the MAS bundle, it knows to which agent the request should be forwarded. Likewise, if an agent needs to use a service from another bundle, it contacts the MAS bundle, which is responsible for contacting the correct bundle, invoking the service and forwarding the result back to the agent. This way, we create a bundle which allows for Jade instances to run behind OSGi implementations in a completely transparent way.

We have hereby detailed our architecture. At a high level, it is composed of components which share information based on Web Services. Each one of these components can then be detailed and looked closer in means of the pieces they are made of: sensors, actuators, MAS, software, etc. The communication inside the components is based on OSGi and R-OSGi open standards, granting extensibility, modularity, dynamics and a logical hierarchical organization of the pieces that make part of each component.

Communications

Let us take a closer look to the challenge of making such different components to work together. This challenge comes not only from the fact of the architecture being distributed but also from the fact that components may be programmed in different languages and even be running in different platforms. There is therefore the need to establish a mean of communication that is possible to use on all the platforms or all the languages that the components can use. More than just choosing the mean of communication, the language used must be specified so that interoperation between components is possible.

As we have stated before, we have chosen the Web Services to implement the communication between components since they are platform independent and work over networks. The information that is shared through Web Services is in XML format and what our Web Services share is FIPA-ACL messages represented in XML [6]. This FIPA standard allows a description of the main content of the message without having to read the content by using concepts like ontology, language or speech-acts. This way, messages can be forwarded and sent to the final agents without the need to check the content.

However, we have defined a way of structuring the actual content of the message in XML. Examples of content are the temperatures in a room or in an entire house (a list of rooms), an aspect of the Electronic Health Record (EHR), the whole EHR or a recommendation coming from the Recommendation System. As an example we have below a simple message from the Recommendation System, asking the house about the values of the temperature and movement on all the rooms of the house:

```
<?xml version="1.0"?>
<fipa-message>
 <act> request </act>
 <msg-param>
  <sender>
   <agent-identifier>
    <name> groupdecision </name>
    <addresses>
     <url>
http://abc.com/groupdecisionwebservice </url>
    </addresses>
   </agent-identifier>
  </sender>
 </msg-param>
 <msg-param>
  <receiver>
```

```
          <agent-identifier>
           <name> house </name>
           <addresses>
            <url> http://def.com/housewebservice </url>
           </addresses>
          </agent-identifier>
         </receiver>
        </msg-param>
        <msg-param>
         <content>
          <sensors room="all"> temperature </sensors>
          <sensors room="all"> movement </sensors>
         </content>
        </msg-param>
        <msg-param>
         <conversation-id> 88273847728729
       </conversation-id>
        </msg-param>
       </fipa-message>
```

As an example of a sequence of communication, we can look at Figure 13. All the process is triggered by the bundle responsible for monitoring the vital signs of the Supported User. This bundle detects an irregular heart beat and warns the House central OSGi, where the MAS is running using R-OSGi. The MAS requests information from the movement sensors in another OSGi and asks again about the cardiac rhythm to the bundle that started the process to ensure that there was no reading error. Having gathered the information, the MAS decides that it cannot do anything to correct the situation and informs the Group Decision sending the anomalous values. This one contacts the Recommendation System which reads all the values of the sensors of the House and generates a recommendation which is then issued back to the Group Decision. After communicating with some more elements (like specialized doctors) and having in consideration the answer from the Recommendation System, two actions are taken: an ambulance is sent to the Home and Lights in the room of the user are turned on.

Dashed arrows represent R-OSGi services being invoked, regular arrows stand for FIPA ACL messages being exchanged through Web Services and circles represent some major processing or communication with local bundles using OSGi. Due to lack of space, this picture is simplified. For example, in the last two lines, a request from the GD to turn on the lights would arrive to the Home central, from there to the OSGi lights implementation bridge bundle and from there to the bundle that control the lights of the room the user is in, which would issue the respective X10 command. The Sensors box also represents all the sensors in general and what really happens is that there is a group of OSGi implementations, each one of them controlling a type of sensors, grouped by room and by floor.

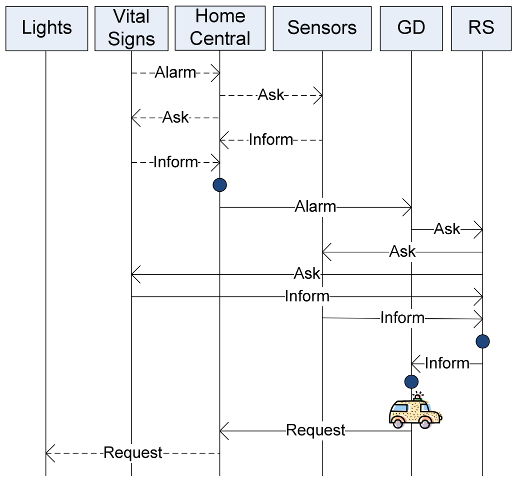


Figure 13. Sample Communication Sequence Diagram

The complete architecture has, therefore, much more components and is much more complex. However, using this approach we can, despite the complexity, create a logical and hierarchical organization that works well as long as each bundle knows the bundles it should directly communicate to. By doing so and defining the messages content structure, we implement the communication methodology needed to give life to the architecture.

Conclusion and Future Work

The new reality in the healthcare sector to allow a dignified care provisioning to all the population in general, and the elderly in particular imposes new approaches to provide specialized services, without delocalizing or messing up with their routines, in a more effective and intelligent way. This chapter describes the VirtualECare project with special incidence on the Group Decision module that supports asynchronous and distributed meetings set up for solving multi-criteria decision problems. The system supports the meeting participants in constructing and sharing ideas and "defends" those ideas in order to reach consensus or majority. To defend his ideas, each participant, should argue for the most interesting alternatives or against the worst alternatives, according to his/her preferences and/or skills, expecting to influence the others' opinions and make them change their own. In future work the argumentation module will allow not only a simple way of justifying opinions, but also a persuasive argumentation in order to allow each element to try influence other through the confrontation of opinions.

Additionally, we are going to apply the presented Knowledge Representation with the respective Quality of its Information to the Group Decision module. Thus, the suggestions/decisions presented by this module, will consider the existence of incomplete information, and, even so, will present a possible way to try and, if possible, resolve the actual problem. Incomplete information may arise from several sources (e.g. unreachable sensors, incomplete Patient Electronic Clinical Profile), but what is important is to be able to measure the quality of the information we have access to and the quality of the ideas presented by the participants, based in factors like reputation, credibility, namely, in the discussion. However, we are certain, that some vital information, if incomplete, may even so, compromise any suggestion/decision but, in the majority of situations, we believe this will not be the case.

We have also presented, based on open standards, a framework to an early deployment of a prototype for the VirtualECare system. In future work, we expect to elaborate on real life scenarios and situations, in order to make the necessary's developments to set a working prototype, that could provide to the population in general, and the elderly, in particular, a certain amount of remote services (e.g., healthcare, entertainment), without delocalizing or messing up with their routines, in a more effective and intelligent way.

Attending to the presented scenarios and a possible ways to make it reality, we present how we may use collaborative networks as a support for different, but interconnected virtual organizations, that could provide to all the population in general, and the elderly, in particular way, a certain amount of remote services (e.g. healthcare, entertainment, learning), without delocalizing or messing up with their routine, in a more effective and intelligent way.

References

1. Ford, E.S., Capewell, S.: Coronary Heart Disease Mortality Among Young Adults in the U.S. From 1980 Through 2002: Concealed Leveling of Mortality Rates. JACC (2007) 2128-2132
2. WHO: Active ageing: towards age-friendly primary health care. World Health Organization (2004)
3. Giráldez, M., Casal, C.: The Role of Ambient Intelligent in the Social Integration of the Elderly. IOS Press (2005)
4. Camarinha-Matos, L.M., Afsarmanesh, H.: Virtual Communities and Elderly Support. WSES (2001)
5. Saranummi, N., Kivisaari, S., Sarkikoski, T., Graafmans, J.: Ageing & Technology - State of the art. Report for the European Commission, Institute for Prospective Studies, Seville, Spain (1996)
6. Costa, R., Novais, P., Machado, J., Alberto, C., Neves, J.: Inter-organization Cooperation for Care of the Elderly. Springer-Verlag, Wuhan, China (2007)
7. Costa, R., Neves, J., Novais, P., Machado, J., Lima, L., Alberto, C.: Intelligent Mixed Reality for the Creation of Ambient Assisted Living, Vol. 4874. Lecture Notes in Artificial Intelligence - Spinger (2007)
8. Sarela, A., Korhonen, I., Lotjonen, J., Sola, M., Myllymaki, M.: IST Vivago® - an intelligent social and remote wellness monitoring system for the elderly. 4th International IEEE EMBS Special Topic Conference on Information Technology Applications in Biomedicine, Birmingham, UK (2003) 362-365
9. Augusto, J.C., McCullah, P., McClelland, V., Walden, J.-A.: Enhanced Healthcare Provision Through Assisted Decision-Making in a Smart Home Environment. 2nd Workshop on Artificial Inteligence Techniques for Ambient Inteligence (2007)
10. Camarinha-Matos, L.M.: Virtual Enterprises and Collaborative Networks (IFIP International Federation for Information Processing). Kluwer Academic Publishers (2004)
11. Camarinha-Matos, L.: New collaborative organizations and their research needs. PRO-VE'03. Kluwer Academic Publishers (2003)
12. Marreiros, G., Santos, R., Ramos, C., Neves, J., Novais, P., Machado, J., Bulas-Cruz, J.: Ambient Intelligence in Emotion Based Ubiquitous Decision Making. Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2007) - 2nd Workshop on Artificial Intelligence Techniques for Ambient Intelligence (AITAmI'07) (2007)

13. Bostrom, R., Anson, R., Clawson, V. (eds.): Group facilitation and group support systems. Macmillan (2003)

14. Schwarz, R.M.: The Skilled Facilitator: Practical Wisdom for Developing Effective Groups. Jossey Bass (1994)

15. Dubs, S., Hayne, S.C.: Distributed facilitation: a concept whose time has come? : Computer Supported Cooperative Work (1992) 314-321

16. Rittel, H., Webber, M.: Dilemmas in a General Theory of Planning. Policy Sciences **4** (1973) 155-169

17. Brito, L., Novais, P., Neves, J.: The logic behind negotiation: from pre-argument reasoning to argument-based negotiaion. In: Plekhanova, V. (ed.): Intelligent Agent Software Engineering. Idea Group Piblishing (2003) 137-159

18. Conklin, J.: The IBIS Manual: A short course in IBIS methodology. GDSS Inc.

19. Conklin, J., Begeman, M.: gIBIS: A Hypertext tool for exploratory policy discussion, Vol. 6 (1988)

20. Hustadt, U.: Do we need the closed-world assumption in knowledge representation? : KI'94 Workshop. Baader, Buchheit, Nutt (eds.), Saarbrüken, Germany (1994)

21. Analide, C.: Antropopatia em Entidades Virtuais, Ph.D. Thesis. Engineering School, Vol. Ph.D. University of Minho, Braga, Portugal (2004)

22. Neves, J.: A Logic Interpreter to Handle Time and Negation in Logic Data Bases. Proceedings of the ACM'84, The Fifth Generation Challenge (1984) 50-54

23. Gelfond, M., Lifschitz, V.: Logic Programs with Classical Negation. Proceedings of the International Conference on Logic Programming (1990)

24. Parsons, S.: Current approaches to handling imperfect information in data and knowledge bases. IEEE Trans. on Knowledge and Data Eng. **8** (1996)

25. Holmlid, S., Björklind, A.: Ambient Intelligence to Go. AmIGo White Paper on mobile intelligent ambience (2003)

26. Initiative, O.S.G.: Osgi Service Platform, Release 3. IOS Press (2003)

27. Chen, K., Gong, L.: Programming Open Service Gateways with Java Embedded Server(TM) Technology. Prentice Hall PTR (2001)