eleventh
international
conference
on
autonomous
agents and
multiagent
systems

AAMAS 2012

4th- 8th June 2012
Valencia

W1 Workshop on Agents Applied in Health Care (AAHC)

VII Workshop on Agents Applied in Health Care, A2HC 2012

held within the *11th International Conference on Autonomous Agents and Multiagent Systems , AAMAS 2012*

Valencia, Spain, June 4th, 2012

Editors:

Antonio Moreno
ITAKA Research Group
Universitat Rovira i Virgili, Tarragona, Spain

Ulises Cortés
Software Department
Technical University of Catalonia (UPC),
Barcelona, Spain

Magí Lluch-Ariet
Informatics Research Institute
Barcelona Science Park, Barcelona, Spain

David Isern
ITAKA Research Group
Universitat Rovira i Virgili, Tarragona, Spain

VII Workshop on Agents Applied in Health Care, A2HC 2012

Multi-agent systems are one of the most exciting research areas in Artificial Intelligence. In the last ten years there has been growing interest in the application of agent-based systems in health care, and it has been argued that the properties of agents fit very well with the usual characteristics of the problems found in health care.

The first specialised workshop on this area was held at Autonomous Agents '2000; several other workshops have followed since then, including three ECAI workshops in 2002, 2004 and 2006. Moreover, a growing European community of researchers interested in the application of intelligent agents in health care emerged as a result of the activities within the AgentCities European project and the AgentLink III Technical Forum Group on Healthcare Applications of Intelligent Agents.

The field is now starting to have some academic maturity, and it may now be a good time for the specialists in the field to meet and report on the results achieved in this area, to discuss the benefits (and drawbacks) that agent-based systems may bring to medical domains, and also to provide a list of the research topics that should be tackled in the near future to make the deployment of health-care agent-based systems a reality.

In this occasion, the VII Workshop on Agents Applied in Health Care will be held in conjunction with the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)

Antonio Moreno
Ulises Cortés
Magí Lluch-Ariet
David Isern

June 2012

Organization

Workshop chairs

A. Moreno, Univ. Rovira i Virgili, Spain

U. Cortés, Technical University of Catalonia, Spain

M. Lluch-Ariet, Informatics Research Institute, Spain

D. Isern, Univ. Rovira i Virgili, Spain

Program Committee

D. Glasspool, Deontics, UK

M. Beer, Sheffield Hallam Univ., UK

F. Grasso, Univ. of Liverpool, UK

A. Ilarramendi, Univ. of the Basque Country, Spain

P. Kostkova, City Univ., UK

L. Lhotska, Czech Technical Univ., Czech Republic

B. López, Univ. of Girona, Spain

M. Schumacher, Univ. of Applied Sciences Western Switzerland, Switzerland

A. Valls, Univ. Rovira i Virgili, Spain

Table of Contents

Performance evaluation of MOSAIC: A Multi Agent System for multilateral exchange agreements of clinical data*

Magí Lluch-Ariet[1,2] magi.lluch@upc.edu
Albert Brugués de la Torre[1] albert.brugues@entel.upc.edu
Josep Pegueroles-Vallés[1] josep.pegueroles@upc.edu

[1]Departament d'Enginyeria Telemàtica
Universitat Politècnica de Catalunya (UPC)
C/ Jordi Girona, 1-3, C3
08034 Barcelona, Catalonia (Spain)

[2]Institut de Recerca en Enginyeria Informàtica
Informatics Research Institute (IRI)
C/ Baldiri Reixac, 4-8
08028 Barcelona, Catalonia (Spain)

ABSTRACT

The understanding of certain data often needs to collect similar data from different places to be analysed and interpreted. Interoperability standards and ontologies, are facilitating the data interchange around the world. However, beyond the existing networks and advances for data transfer, data sharing protocols to support multilateral agreements are useful to exploit the knowledge of distributed Data Warehouses. The access to a certain data set in a federated Data Warehouse may have the constrain of delivering another data set. When bilateral agreements between two nodes of a network are not enough to solve the constrains for accessing to a certain data set, multilateral agreements for data exchange are needed.

We present the implementation of *MOSAIC* and evaluate how multilateral agreements increase the percentage of the data collected by a single node from the total amount of the data available in the network. Different strategies to reduce the number of messages needed to achieve an agreement are considered (random selection of the path and path selection based on the dataset size).

Results show that *MOSAIC* significantly improve the percentage of data collected from approximately the 30% with bilateral agreements to almost all data available in the network with multilateral ones.

Categories and Subject Descriptors

C.2.2 [**Network Protocols**]; C.2.4 [**Distributed Systems**]: Distributed databases; I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems; H.3.4 [**Systems and Software**]: Distributed Systems; H.3.5 [**Online Information Services**]: Data sharing

General Terms

Algorithms

*For A2HC

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), June, 4–8, 2012, Valencia, Spain.

Keywords

Data Sharing, Clinical Data Sharing, Data Exchange, Network Protocol, Multi-Agent System, Data Warehouse, Federated Data Warehouse, MOSAIC

1. INTRODUCTION

Worldwide collaboration is a fact in most areas of activity and often implies some exchange that is facilitated when the items to transfer correspond to knowledge or information, easily transmitted digitally. Local repositories of data are growing and growing and the search of valuable data becomes very complex or impossible to be managed manually in certain frameworks.

When the access to the information worldwide distributed is important in order to take some decision and when the information is so specialised and formalised that its understanding for a specialist needs little processing, is when providing mechanisms to facilitate the access to remote data sets becomes relevant.

Healthcare is a framework where all these characteristics apply, and the following two scenarios illustrate the potential interest of the MOSAIC system.

1.1 Professional cooperation

Clinicians often need to compare the information collected from the experiments performed to their patients with information from similar patients in other places. This is needed for accurate diagnosis, theragnosis, effective management of the diseases and efficient use of drugs.

Ethical and legal regulations that apply to personal data, and the associated data access authorisations to be provided by the ethical committees, must be integrated in any negotiation process for data exchange. Under the assumption that the access rights to a certain dataset are given, a clinician may also add some additional constrain and give access to the data only if another dataset is given. However, bilateral agreements between two clinical centres will not always solve those constrains and involving a set of centres in multilateral agreements for data exchange would also increase the amount of data potentially accessible in the network.

1.2 Social networks

We can imagine a near future with a considerable number of individuals with their own Electronic Health Record (EHR) in their healthcare ID cards including their genotype. That information will be preserved, protected and regulated

by law as personal and private information. Nowadays, most data of genetic sequences is collected managed and stored in research labs for scientific purposes, but as the next generation sequencing technology becomes more affordable, beside those big data repositories, new distributed databases are appearing and in a near future a significant number of this information will be hosted and managed by every individual, and some of them will be reluctant to allow its storage and use in a database. In such scenario people may what to share its own data only if this brings some benefit to them. Someone may be interested to explore the network to find individuals with similar phenotypes, suffering similar illness to learn from their experience and treatment evolution.

Similarly to the previous scenario, a person may give access to his personal information only if some other information is given by someone else, and bilateral agreements between two persons may not always solve those constrains. Involving a group of people in multilateral agreements for data exchange would increase the amount of data potentially accessible in the network.

1.3 State ot the art

Well established interoperability standards (DICOM [15, 17], HL7 [8] or ISO/EN 13606 [7]), clinical ontologies, systems like caCORE [10], Electronic Health Record [9], and Multi-Agent systems [6, 2, 14] are facilitating the data transfer and interchange between clinical centres around the world. Nevertheless, it is still difficult to find and get access to the best dataset for a certain purpose.

An example of a federated Datawarehouse and its associated Decision Support System is the HealthAgents project [11, 5], that aims to build a system to manage a network of clinical centres for the brain tumour diagnosis. This project was focused on collecting data for building classifiers that would be used during the diagnose process, but did not address the exchange of data between the nodes.

The problem of solving multilateral agreements can be mapped to the problem of finding the shortest path in a complex network, that could be rapidly solved using the Dijkstra algorithm [4], but this is possible only if the links between the nodes are known and the topology of the whole network available in a centralised place. In many scenarios neither the information of the network topology is available in a single place nor this information is complete. As an example of this, a clinician may accept to publish the reference of which datasets are available from his local repository, but the specific constrains to give access to them may not be informed before a explicit data access request from a specific centre is received. Thus, a centralised approach to solve this problem is not feasible and distributed and dynamic mechanisms for the exploration of the paths associated to possible multilateral agreements are needed.

1.4 Scenario evaluation

The scenario evaluation corresponds to a set of nodes (cities) hosting each of them a number of datasets with clinical cases of brain tumour (see Table1). While some datasets are freely offered to the network without any restriction, most of them have a constrain associated, requiring the delivery of some other dataset from some other node.

In this article, we first, provide the details of the implementation of *MOSAIC* for the finding of paths involving a set of nodes that all together can participate in a multilateral agreement for data exchange, according to the design published in [12]. Second, we show the evaluation results of its execution.

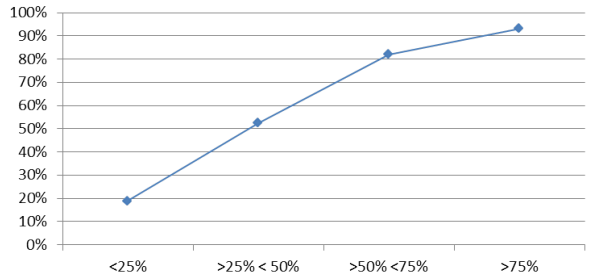

Figure 1: Average of the percentage of cases collected by the nodes after bilateral exchanges.

On the one hand, nodes with a large number of cases covering most of the data types will have a higher chance to directly solve a possible constrain and achieve bilateral agreements. On the other hand, those nodes with a reduced number of cases in their datasets will likely need multilateral agreements to get the data desired from the network. The system presented here is specially useful for those nodes with less chances to achieve bilateral agreements. The figures corresponding to this evidence are shown in Fig. 1 generated after running the *MOSAIC* protocol in the evaluation scenario of the worldwide network. The total set of nodes has been grouped in 4 categories according to the size of their datasets (e.g. "<25%" indicates the category of the set of nodes with a number of cases in their datasets, minor than the 25% of the average size of all datasets in the network).

In this section we have introduced the system developed, the justification why it is needed, and the specific scenario evaluation where it has been deployed. In section 2, the *MOSAIC* components and the negotiation process for the multilateral data exchange is explained. In section 3, we explain the details of the implementation and the path selection for the network exploration. In section 4, we present the results and findings after analysing the results of the protocol execution in the proposed scenario. In section 5, we summarise the main achievements and finally, in section 6 we outline the future work to be done.

2. THE MOSAIC SYSTEM

2.1 The MOSAIC components

The MOSAIC System is composed by a set of interconnected nodes each one with its associated Data Mart and the Agent Platform to host the following Agents:

- **Multicast Contributor Agent (MCC)**. Activated by the user to offer a certain dataset to the network, with or without constrains.
- **Unicast Contributor Agent (UCC)**. Activated by the MCC to negotiate a specific data access request sent by a MCP.
- **Multicast Petitioner Agent (MCP)**. This Agent is activated by the user or by a Unicast Petitioner Agent. The user lauches it in order to explore the network

Table 1: Dataset of the scenario evaluation. Brain tumors by major histology groupings

Brain and CNS Tumors			Worldwide network 2.852 Nodes		USA network 205 Nodes	
Class	Histology	Types	Datasets	Cases	Datasets	Cases
A	Tumors of Neuroepithelial Tissue	A1-A17	8.203	374.580	859	41.360
B	Tumors of Cranial and Spinal Nerves	B1	2.145	114.680	201	12.750
C	Tumors of Meninges	C1-C3	3.135	507.800	238	54.380
D	Lymphomas and Hematopoietic Neoplasms	D1	759	23.610	95	2.690
E	Germ Cell Tumors and Cysts	E1	124	2.310	14	240
F	Tumors of Sellar Region	F1-F2	2.947	188.060	227	20.360
G	Local Extensions from Regional Tumors	G1	5	50	1	10
H	Unclassified Tumors	H1-H2	1.584	58.980	189	6.720

looking for a certain data set. The UCP launches it in order to solve a constrain from a UCC when the dataset requested is not available at the node of the UCP.

- **Unicast Petitioner Agent (UCP)**. Activated by the MCP in order to negotiate a specific data access request with a UCC.
- **Yellow Pages Agent (YP)**. This Agent provides the directory service and hosts the list references of MCC active in the network.

To support the interaction between the agents and the *MOSAIC* users, the system architecture contains:

- The **MOSAIC Dashboard** to facilitate the activation of new data set requests, the visualisation of all the possible agreements, the selection of those chosen by the user, and the acceptance or rejection of constrains at user level required to give the data access rights to certain data sets.
- The **MOSAIC Manager**. A tool to set up the parameters of the protocol and to shape the behavior of the agents according to the user's preferences.

2.2 The MOSAIC negotiation process

From the launch of a data set request by the User to the data delivery and all the corresponding intermediate steps to solve the possible constrains, the *MOSAIC* protocol follows a process with the following five stages:

- **Stage 1: Network exploration**. After the activation of a MCP the process to find paths than connect the requesting node with the ones hosting the desired data starts. This exploration ends with the identification of a set of nodes connected with the initiator (directly or with intermediate connexions with other nodes).
- **Stage 2: Agreement proposal notification**. From the leaf to the initiating MCP. Every agent participating in a successful path will notify to its creator and at the end of this stage the initial MCP receives the existence of all possible agreements for the data exchange.
- **Stage 3: Agreement selection and notification**. The MCP will select a path or a set of paths and notify this decision to all the agents involved, considering among other criteria to avoid overlapping agreements that solve the access to the same dataset of the same MCC through different paths.

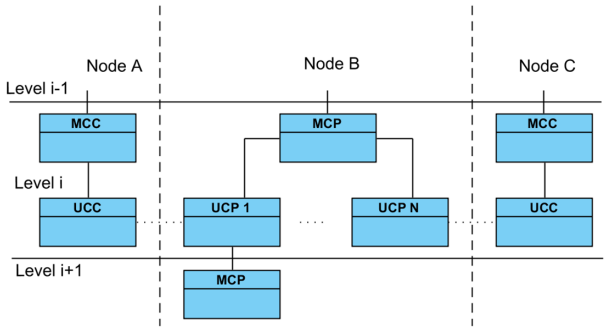

Figure 2: The MOSAIC Agents, their dependency and interactions during the execution of the protocol.

- **Stage 4: Data transfer**. After receiving the notification that a possible agreement is selected, the data exchange between all the nodes starts. This may end with a complete and successful data exchange or with some failure by some node. All the UCP waiting to receive data will send an ACK to their MCP after receiving the data or an NACK in case of failure of the data receiption. The ACK (or NACK) is transmitted link to link until arriving to the main MCP in the top of the path.
- **Stage 5: Transaction completion**. After receiving all the ACK from all the nodes involved in the agreement, the initiating MCP will send a COMMIT to all the Agents. In case some ACK is not received or a NACK is transmitted by some Agent, the MCP will send a ROLLBACK message to all the nodes. Only after the reception of a COMMIT the nodes will have the authorisation to use the data received. In case the transaction is aborted with a ROLLBACK, none of the nodes of an agreement that received some data are authorised to use it.

Fig. 2 shows the dependences and relationships between the *MOSAIC* agents. Every link in a multilateral agreement is composed by the two pairs of MCP-UCP and MCC-UCC. During the negotiation process the agents generate a number of messages and those that correspond to the communications between different nodes involved in a multilateral agreement are the following:

$$M = MP(4UP + 2)$$

Where MP and UP are the number of MCP and UCP respectively, involved in the agreement. The four messages per UCP correspond to the following:

- **MCP → MCC**: Dataset request
- **UCC → UCP**: Notification of the constrain
- **UCP → UCC**: Constrain delivery
- **UCC → UCP**: Acceptance of the agreement

And the two messages per MCP correspond to:

- **MCP → YP**: Request from the MCP to the YP asking for the MCC offering the desired dataset
- **YP → MCP**: Answer from the YP to the MCP with the list of references of MCC available

This measure is useful for the assessment of the communication efficiency in terms of number of messages transmitted in the network, and analysed in section 4.

2.3 Network exploration

In this stage of the *MOSAIC* process, the MCP asks the YP to obtain the list of MCC to whom the data access requests can be addressed. The reference of the MCC delivered by the YP are those hosting a dataset of the type requested by the MCP. For each MCC three situations may arise:

- **No constrain** The MCC offers the requested data set with no constrains to fulfill through a UCC. The UCP receives the notification of the data set availability and notifies this to the MCP.
- **Constrain solved locally** The constrain of the MCC can be solved locally at the requesting node of the MCP. The UCP asks to its MCP to look for the MCC active in its node in order to collect the data from its DataMart. The UCP sends the notification of the dataset availability at the UCC and after the potential fulfillment of the constrain, the UCC sends the agreement for the possible dataset transfer initially requested to the UCP. Both UCC and UCP notify to their MCC and MCP their agreement for the potential exchange of the corresponding datasets.
- **Constrain to be solved externally** The constrain of the MCC can not be solved locally at the requesting node of the MCP. If the length of the path does not exceed the limit (monitored through a Time To Live -TTL- parameter) The UCP launches a new MCP to look for the data set needed in order to solve the constrain.

A node and a MCC can take part more than once in a path of a multilateral agreement, but a special case occurs when in order to solve a constrain of a MCC the subsequent activations of new MCP results in a new request to the same MCC. If the request comes from a MCP "child" (belonging to the same branch), the MCC will decide to activate the UCC without any constrain and thus, deliver its dataset without receiving any dataset in advance (see Figure 3). Doing so, after completing the delivery of the other datasets in the path links the MCC will receive the dataset of its constrain

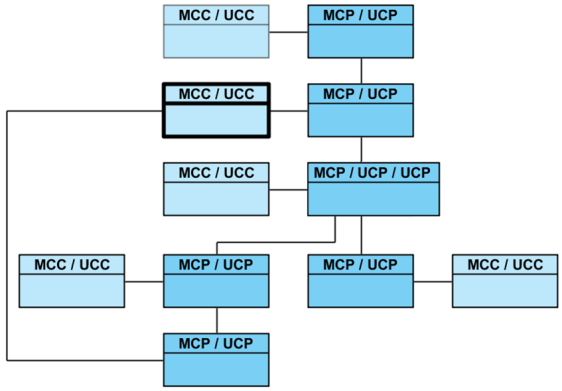

Figure 3: Example of a path in the network exploration where a MCC (in bold) will deliver its dataset without solving its constrain after identifying a loop.

from the first MCP of the branch that started negotiations with him.

The exploration of the network includes the selection of the paths. When a MCP receives the set of MCC candidate, selects a subset of them to continue the network exploration. The criteria of this selection chosen in this version of the protocol is the size of the dataset hosted at the MCC node. This criteria has been compared with the random selection of a MCC among the list of candidates and the results are described in the evaluation of the protocol in section 4.

The *MOSAIC Manager* permits to adjust the network exploration according to the user preferences and the specific density, inclusiveness, or degree distribution of the network.

When a MCP does not find any MCC with the data set needed to fulfill a constrain, stops the exploration and notifies to its UCP creator the failure of the path in its attempt to find a multilateral agreement.

The network exploration concludes when all the paths have: i) conclude successfully a possible multilateral agreement or ii) failed in the agreement exploration or iii) its length arrived to the TTL limit.

3. THE MOSAIC IMPLEMENTATION

The interaction between the actors of *MOSAIC* respects the following principles: i) The Users of the protocol interact with the Multicast Agents, ii) Unicast Agents are created by Multicast Agents to negotiate every possible data exchange between two nodes, and iii) Multicast Agents talk with Multicast Agents, Unicast agents with Unicast Agents, Petitioner Agents with Petitioner Agents, and Contributor Agents with Contributor Agents. Direct communications between a MCP and a UCC or between a MCC and a UCP are avoided.

Two important aspects of the implementation correspond to i) the way that a path of a possible agreement is created and propagated and ii) the way a loop is detected.

3.1 Agreement paths

After the activation of a new request by the user a *Request* object is created. An instance of this object will be linked to every UCP and includes i) the Id of the requesting node, ii) the Id of the first MCP Agent of the negotiation chain, and iii) the Id of the negotiating branch. The value of the Id of

the negotiating branch corresponds to a list of numbers that increases at every step of the path creation. When a MCP is launched by another MCP it receives from its creator the *Request* object and adds to the branch Id a new number. Doing so the *Request* object will contains the information needed to create the agreement paths.

A UCP arrives to the end of a path candidate to solve a multilateral agreement, when it receives the requested data from its UCC without the need to launch any other MCP. Then, it creates an object that will represent the negotiation path to whom the UCP belongs to. This object is propagated to the higher levels of the Petitioners chain up to the MCP that initiated the request. During this bottom up process of transferring the agreement path candidate, all the Petitioners, at every link of the path, add to that object the relevant information and reference of the nodes to which there is a possible agreement, corresponding to the nodes where the MCC participating in the negotiation process with every MCP are hosted. At the end of the process for the network exploration, the MCP that initiated the request receives, for every dataset of interest, the set of negotiation paths corresponding each of them to a possible multilateral agreement to reach that dataset. At that point, the MCP decides which negotiation paths to select among all candidates. An initial selection is among the paths that arrive to the same dataset, but the MCP may also decide to execute only a subset of all the remaining negotiation path candidates, depending on other criteria (e.g. cost or reputation).

3.2 Loop detection

Each branch of the $Petitions\,Tree$ is build during the network exploration. Every branch is identified with a *request identifier* corresponding to an array where each of its elements represents the participation of a Petitioner in the brach. It is important to note that a MCP will belong to more than one branch when i) it has more than one UCP exploring different options of agreement or ii) there is some MCP in the lower levels of its path with this situation (managing more than one UCP).

A new request received by a MCC is processed and compared with all the other active requests managed by the MCC. If all the elements of the array of some request identifier active in the MCC is equal to the first elements of the request identifier of the new request received means, that the request comes from the same branch of that already active request at the MCC and a loop is identified. In that case, the associated UCC will be created without any constrain.

3.3 System simulations

Besides the core of the $MOSAIC$ system and its agents, and in order to facilitate its evaluation, a specific component for simulations has been build. This component reads a DB that represents the topology of the network of the evaluation scenario, facilitates the massive activation of the MCC and MCP simulating the users and creates a set of log files to allow the analysis of the results of the process. The results shown in this article are based on first, the activation of the MCC for the datasets with same cases available and second, the activation of a request (or MCP) for every possible dataset, by every node (or city). This corresponds to 2.852 x 28 = 79.856 requests for the whole worldwide network and to 205 x 28 = 5.740 requests for the USA network.

With the purpose of simplifying the evaluation of the protocol and to focus to its core features, the data transfer between Agents is merely a simulation, and the integration of interoperability standards is not implemented, yet.

3.4 Pseudo-code

The MCP has been implemented according to Algorithm 1, the MCC is presented in Algorithm 2, the UCC in Algorithm 3, and the UCP implementation is presented in Algorithm 4. In order to clarify the process and to highlight only the most important features of the protocol, the pseudocode presented here merges the steps of stages 2 to 5 and after an agreement, the dataset is directly transfered to the requesting agent.

Algorithm 1 Multicast Petitioner Agent (MCP)

Inputs

ResourceRequested from $User$ or UCP
$NegotiationAgreement$ from UCP
ResourceDataset from UCP
MCC from $YellowPages$ (YP)

```
1: Ask YP for MCC hosting the ResourceRequested
2: Collect MCC compatible from YellowPages
3: Select MCC to negotiate
4: for all MCC selected do
5:    Create UCP(ResourceRequested)
6:    Ask the UCP to start the negotiation
7:    if NegotiationAgreement = TRUE then
8:       Collect ResourceDataset from UCP
9:       Send ResourceDataset to the User or UCP
10:   end if
11: end for
```

Algorithm 2 Multicast Contributor Agent (MCC)

Inputs

$ResourceOffered$ from the $User$
$Constrain$ from the $User$
Request from the MCP
$NegotiationAgreement$ from the UCC
$ConstrainDataset$ from the UCC

```
1: Add MCC to the YellowPages (YP)
2: while User does not stop the MCC do
3:    Get Request from some MCP
4:    if Request = ResourceOffered then
5:       if Child-Loop detected then
6:          Create UCC(Request, NUL)
7:       else
8:          Create UCC(Request, Constrain)
9:       end if
10:      Ask the UCC to start the negotiation
11:      if NegotiationAgreement = TRUE then
12:         Collect ConstrainDataset from UCC
13:      end if
14:   end if
15:   Remove UCC
16: end while
17: Remove MCC from the YP
```

Algorithm 3 Unicast Contributor Agent (UCC)

Inputs

Request from UCP
Constrain from MCC
$ResourceOffered$ from MCC
ConstrainDataset from UCP
ConstrainSolved from UCP

```
1: if Constrain = none then
2:    Send ResourceOffered to UCP
3:    NegotiationAgreement ← TRUE
4: else
5:    Ask the UCP to solve the constrain
6:    if ConstrainSolved = TRUE then
7:       Collect ConstrainDataset from UCP
8:       Send ConstrainDataset to MCC
9:       NegotiationAgreement ← TRUE
10:   else
11:      NegotiationAgreement ← FALSE
12:   end if
13: end if
14: return NegotiationAgreement
```

Algorithm 4 Unicast Petitioner Agent (UCP)

Inputs

ResourceRequested from MCP
Constrain from UCC
ConstrainDataset from MCC

```
1: Ask the UCC to send the ResourceRequested
2: if Constrain ≠ NUL then
3:    Search MCC in the Node to solve the constrain
4:    if MCC ≠ NUL then
5:       ConstrainSolved ← TRUE
6:       Get ConstrainDataset from MCC
7:       Send ConstrainDataset to the UCC
8:    else
9:       Create MCP to look for the ConstrainDataset
10:      if ConstrainDataset found then
11:         ConstrainSolved ← TRUE
12:         Send ConstrainDataset to the UCC
13:      else
14:         ConstrainSolved ← FALSE
15:         Notify failure to solve the constrain to the UCC
16:      end if
17:   end if
18: end if
19: if Constrain = NUL or ConstrainSolved then
20:    Collect ResourceRequested from the UCC
21:    Send ResourceRequested to MCP
22:    NegotationAgreement ← TRUE
23: else
24:    NegotationAgreement ← FALSE
25: end if
26: return NegotationAgreement
```

4. PERFORMANCE EVALUATION

4.1 The Scenario evaluation

Brain tumours have a considerable number of types and subtypes (see table 2). It is common that a part from the clinical information of the patient, other important data is obtained from the Magnetic Resonance Imaging and Spectroscopy. However, the gold standard for their classification [13] is based on the histopathological analysis of a tumour tissue sample obtained from a biopsy. Nevertheless, for several reasons, this histopathological analysis can not be obtained in a percentage of the cases, and the tumour classification from clinical data, MRI and MRS, is sometimes difficult for clinicians and radiologists. In order to facilitate the diagnose, comparing the information collected from a patient, with similar cases from other patients already diagnosed, is important. Moreover, the knowledge of the effect of certain therapies to other patients with similar profiles may help the clinician to provide a more effective and efficient treatment to his or her patients. The $MOSAIC$ system can help both diagnosis and theragnosis of brain tumours by facilitating the multilateral agreements for data exchange.

In this framework, the scenario evaluation used to test the $MOSAIC$ system, is composed by a set of nodes each of them with a number of datasets corresponding to a subset of brain tumour types. For each dataset each node activates a MCC. Every MCC is associated to a constrain corresponding to a certain tumour data type randomly selected from all types or - with the same probability as any data type - to a void constrain, in which case the MCC freely offers its dataset to any MCP.

The Central Brain Tumor Registry of the United States (CBTRUS) [3] provides statistics of the incidence rates of the different tumour types in United States. This distribution has been used to assign the number of cases of each type to a number of cities, adjusted to their respective population. Two datasets have been created: One with 2.852 nodes corresponding to the main cities around the world, hosting 18.902 data sets; and another with 205 cities from the United States, with 1.824 datasets in total. The evaluation of $MOSAIC$ is performed in this simulated, but realistic scenario.

4.2 Evaluation results

A first evaluation of the $MOSAIC$ protocol is a cross validation to check that it works properly. For this, algorithm 5 has been created. It scans the network of 2.852 cities and their 18.902 data sets seeking for all possible bilateral agreements This algorithm generates as output a matrix with the figures corresponding to all the cases collected by each node for every data type after the bilateral data exchanges. These figures have been compared with those obtained by the execution of the $MOSAIC$ protocol with the Time to Live parameter set to 1, forcing that the maximum length of every multilateral agreement is limited to 2 nodes.

The results obtained in both cases are exactly the same (see Figure 4).

The second evaluation is to prove the main goal of the $MOSAIC$ system which is to overcome the amount of data that can be exchanged with bilateral agreements and collect as much data as possible from the network by achieving as much data exchange agreements as possible. The results obtained strongly depend on the parameters of the protocol,

Table 2: Datasets of the scenario evaluation

Type	Histology
A1	Pilocytic astrocytoma
A2	Protoplasmic and fibrillary astrocytoma
A3	Anaplastic astrocytoma
A4	Unique astrocytoma variants
A5	Astrocytoma, NOS
A6	Glioblastoma
A7	Oligodendroglioma
A8	Anaplastic oligodendroglioma
A9	Ependymoma/anaplastic ependymoma
A10	Ependymoma variants
A11	Mixed glioma
A12	Glioma malignant, NOS
A13	Choroid plexus
A14	Neuroepithelial
A15	Non-malignant and malignant neuronal/glial
A16	Pineal parenchymal
A17	Embryonal/primitive/medulloblastoma
B1	Nerve sheath, non-malignant and malignant
C1	Meningioma
C2	Other mesenchymal, non-malignant and malig.
C3	Hemangioblastoma
D1	Lymphoma
E1	Germ cell tumors, cysts and heterotopias
F1	Pituitary
F2	Craniopharyngioma
G1	Chordoma/chondrosarcoma
H1	Hemangioma
H2	Neoplasm, unspecified

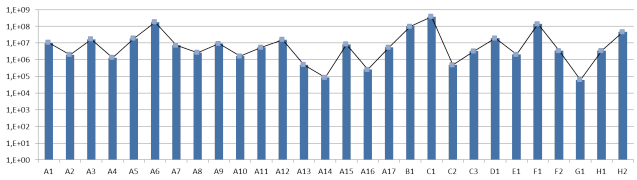

Figure 4: Total number of cases collected from bilateral agreements using *MOSAIC* with TTL=1 (bars), and algorithm 5 (line). In both cases, the result is the same.

Algorithm 5 Search for bilateral exchange agreements

```
1: for i = 1 to numCities do
2:   for j = 1 to numDataTypes do
3:     for k = 1 to numCities do
4:       if i ≠ k then
5:         if Dataset[k, j] ≠ 0 then
6:           if constrain[k, j] = nul then
7:             collect Dataset[k, j] for node i
8:           else if constrain[k, j] available in node i
             then
9:             solve constrain from data in node i
10:            collect Dataset[k, j] for node i
11:          end if
12:        end if
13:      end if
14:    end for
15:  end for
16: end for
```

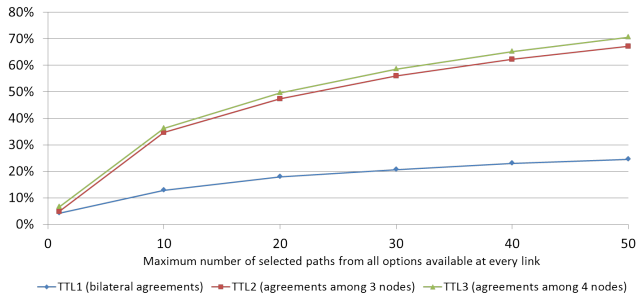

Figure 5: Percentage of cases collected from the total number available in the network with different values of TTL and size of the selected path set.

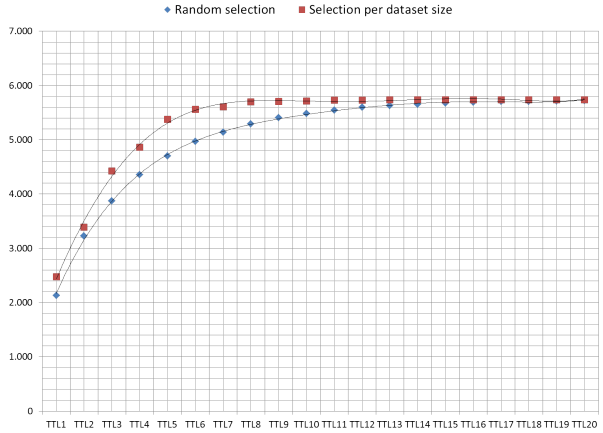

Figure 6: Total number of agreements using *MOSAIC* with different TTL values.

namely its TTL and the number of branches selected from all the paths available during the network exploration. Figure 5 shows the results with three TTL (1: bilateral agreements, 2: agreements among 3 nodes, and 3: agreements among 4 nodes) and with a range selection of paths starting from 1 (only exploring a single MCC from all available) to 50. The percentages of data collected show a steady increase when the selection of the number of possible paths increases, and while the improvement from TTL1 to TTL2 is significant, the increase from TTL2 to TTL3 is limited.

Finally, the third evaluation refers to the optimisation process for the network exploration through the intelligent selection of the paths to follow. As indicated in section 3, the MCP receives the list of MCC compatible from the Yellow Pages and in order to avoid unmanageable network explorations the MCP has to decide which to select and which to discard. Two cases have been evaluated. One selects a MCC randomly from those available, and the other selects the MCC with the biggest dataset. The two cases have been tested using the database of 205 cities from USA with 1.824 datasets in total.

Figures 6 and 7 show the improvement in the number of agreements, when selecting the path to follow during the network exploration according to the size of the MCC dataset, instead of a random selection among the MCC available. When a MCP child (belonging to the same branch) needs

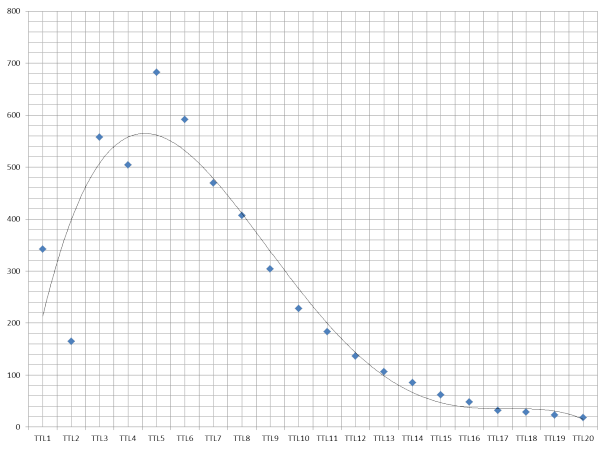

Figure 7: Increase in the number of agreements using a branch selection per node size instead of a random selection.

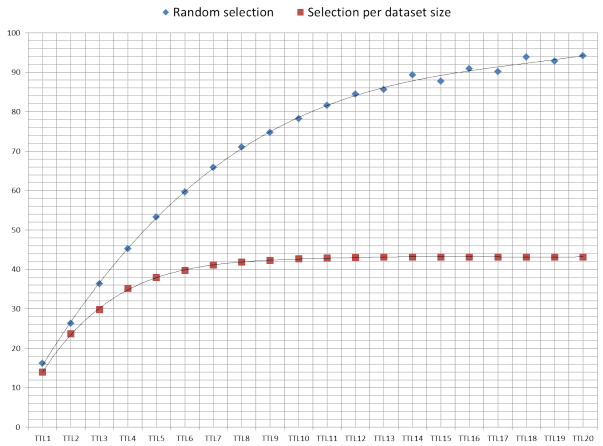

Figure 8: Comparative of the average number of messages needed to achieve an agreement, between the two branch selection strategies.

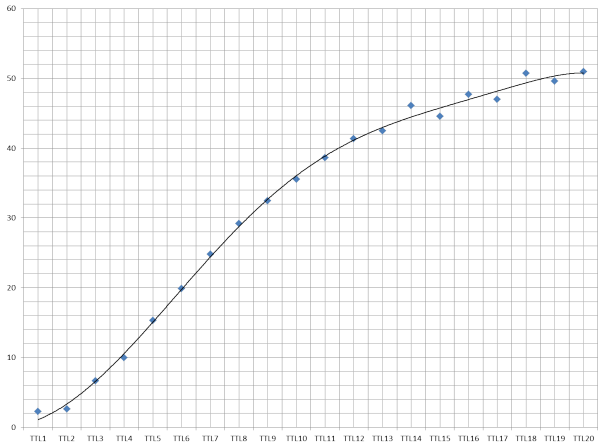

Figure 9: Number of messages saved per agreement in average when using the path selection per dataset size instead of a random selection.

to solve some constrain to obtain the desired dataset, it is more likely to have a MCC in the higher levels of its branch that can solve it if the MCC's dataset is bigger than if it is smaller. In those cases, the MCC that receives a data access request from some child node (belonging to the same branch) decides to offer the requested dataset to solve the constrain as this loop will benefit the overall multilateral agreement and the MCC will also get the desired dataset initially included as a constrain.

Figures 8 and 9 show the reduction in the number of messages transmited over the network and needed to achieve an agreement, comparing the selection criteria of the MCC between the strategy based on the dataset size and the random selection.

5. CONCLUSIONS

It has been demonstrated that the multilateral agreements among a set of nodes increase significantly the amount of data accessible in a network compared with the amount of data that can be collected from bilateral agreements.

Besides, the need of a distributed process to support the achievement of multilateral agreements has been justified for the lack of global knowledge of the network topology derived from the reluctance to publish certain information in a centralised repository. The use of Agents has facilitated to model the negotiation process required by the actors of this system and seams a natural way to implement the protocol.

It has been proved that the strategy to select the path to follow during the exploration of the network has implications in the number of agreements achieved among the nodes. For this, two criteria have been tested: i) A random selection and ii) A selection based on the Dataset size. Finally, it has been demonstrated that the total number of agreements among the nodes achieve better marks when the path selection is based on the dataset size.

6. FUTURE WORK

The research presented here is being extended or it is planned to be extended in the following aspects:

- **Semantic representation**. Both datasets and constrains represented using OWL.
- **Constrains enrichment**. A more natural representation of the possible constrains will be based on a boolean expression composed by a set of clauses, some of them related to the delivery of a combination of certain datatsets (not only a single one) and others related to the acceptance or rejection of certain top level conditions for the data access by the user.
- **Core implementation**. The optimisation of the code to allow wider and deeper path explorations of the network in a reasonable time and the visualisation of the protocol results in a web based interface.
- **Security and privacy**. Data disclosure protection, attacks prevention, authenticity, and other features of privacy and security are issues that will be integrated in the protocol with the deployment of previous research in the field [20, 18] and their adaptation to this specific scenario.

- **System deployment and evaluation in different scenarios**. It is expected that the results of the protocol will differ significantly depending on the specific scenario and characteristics of the network. Therefore, it is also planned to adapt the behavior of the Agents to different frameworks and to identify which strategies are the best for each case and specifically the best balance between path length and branch selection wide (number of branches to explore among all the possible). The next scenario evaluation candidate to test the MOSAIC system is 1.000 *Genomes* [1].
- **Intelligent exploration**. Increase the intelligence of the path selection by including more advanced indicators (e.g. reputation, user similarity and cost) considering also previous research in the areas of agent trust, argumentation and reasoning [19, 16].

The system is being developed under LGPL licences and in a near future it will be uploaded in a public repository in order to facilitate its evolution, integrating contributions from the scientific community.

7. ACKNOWLEDGMENTS

We would like to thanks our colleagues from the Telematics Engineering department at the UPC, for the inspiring discussions we had with them regarding the content of this paper. Also to the Central Brain Tumor Registry of the United States for their excellent work in the creation of the CBTRUS database and the possibility to use it in this research. Finally, we would like also to tanks Dr. Paul Flicek for his initial interest to explore a possible evaluation of the *MOSAIC* system in the repositories managed at the European Bioinformatics Institute.

This research has been partially funded by the AGAUR agency of the Catalan Government through the grant with reference number 2010-TEM-88 and by the project TAMESIS (TEC2011-22746).

8. REFERENCES

[1] D. L. Altshuler et al. A map of human genome variation from population-scale sequencing. *Nature*, 467(7319):1061–1073, 2010.

[2] R. Annicchiarico, U. Cortés, and C. Urdiales. *Agent Technology and e-Health*. Whitestein Series in Software Agent Technologies and Autonomic Computing. Birkhuser Basel, first edition, 2008.

[3] Central Brain Tumor Registry of the United States. Statistical report: Primary brain tumors in the united states (2004-2008). web site. http://www.cbtrus.org/ (Last access: 29-02-2012).

[4] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959. 10.1007/BF01386390.

[5] F. Estanyol, X. R. Palou, R. Roset, M. Lurgi, M. Mier, and M. Lluch-Ariet. A web-accessible distributed data warehouse for brain tumour diagnosis. *Knowledge Eng. Review*, 26(3):329–351, 2011.

[6] D. Isern, D. Sánchez, and A. Moreno. Agents applied in health care: A review. *International Journal of Medical Informatics*, 79(3):145 – 166, 2010.

[7] ISO 13606. Electronic health record communication – part 1: Reference model, 2008.

[8] ISO/HL7 27931. HL7 version 3 - Reference information model, 2006.

[9] ISO/TR 20514. Electronic health record – definition, scope and context, 2005.

[10] G. A. Komatsoulis, D. B. Warzela, F. W. Hartela, K. Shanbhaga, R. Chilukuric, G. Fragosoa, S. de Coronadoa, D. M. Reevesa, J. B. Hadfieldа, C. Ludetb, and P. A. Covitza. cacore version 3: Implementation of a model driven, service-oriented architecture for semantic interoperability. *Journal of Biomedical Informatics*, 41:106–123, 2008. DOI: 10.1016/j.jbi.2007.03.009.

[11] M. Lluch-Ariet, F. Estanyol, M. Mier, C. Delgado, H. González-Vélez, T. Dalmas, M. Robles, C. Sáez, J. Vicente, S. V. Huffel, J. Luts, C. Arús, A. P. C. Silveira, M. Julià-Sapé, A. Peet, A. Gibb, Y. Sun, B. Celda, M. C. M. Bisbal, G. Valsecchi, D. Dupplaw, B. Hu, and P. Lewis. *On the Implementation of HealthAgents : Agent-Based Brain Tumour Diagnosis*. Whitestein Series in Software Agent Technologies and Autonomic Computing. Birkhuser Basel, first edition, 2008.

[12] M. Lluch-Ariet and J. Pegueroles-Vallés. The mosaic system - a clinical data exchange system with multilateral agreement support. Casablanca, 12/2010 2010. 3rd International ICST Conference on Electronic Healthcare for the 21st century, 3rd International ICST Conference on Electronic Healthcare for the 21st century.

[13] D. Louis, H. Ohgaki, O. Wiestler, W. Cavenee, P. Burger, A. Jouvet, B. Scheithauer, and P. Kleihues. The 2007 who classification of tumours of the central nervous system. *Acta Neuropathologica*, 114:97–109, 2007. 10.1007/s00401-007-0243-4.

[14] E. Merelli, G. Armano, N. Cannata, F. Corradini, M. d'Inverno, A. Doms, P. Lord, A. Martin, L. Milanesi, S. Moller, M. Schroeder, and M. Luck. Agents in bioinformatics, computational and systems biology. *Brief Bioinform*, 8(1):45–59, 2007.

[15] H. Oosterwijk. *DICOM Basics*. O Tech, third edition, 2008.

[16] S. Parsons, Y. Tang, E. Sklar, P. McBurney, and K. Cai. Argumentation-based reasoning in agents with varying degrees of trust. In *AAMAS*, pages 879–886, 2011.

[17] O. S. Pianykh. Digital imaging and communications in medicine (dicom), 2008.

[18] D. Rebollo-Monedero and J. Forné. How do we measure privacy. *Upgrade*, pages 53–58, 01/2010 2010.

[19] C. Sierra and J. Debenham. Information-based reputation. pages 5–19, Gargonza, Italy, 2009.

[20] J. Tomàs-Buliart, M. Fernández, and M. Soriano. Protection of mobile agents execution using a modified self-validating branch-based software watermarking with external sentinel. 5508:287–294, 2009.

APPENDIX

A. THE SCENARIO EVALUATION DB

The main data base created for the simulations is composed by a network of 2.852 nodes corresponding to cites worldwide distributed with 18.902 datasets in total.

The CBTRUS Statistical report provides the figures of the total number of Brain and CNS tumours in USA by major histology groupings (see table 3) and a simulated number of cases per node has been calculated according to the population of every city and the resulting proportional figure considering the USA population and the number of tumour cases from the CBTRUS DB. Table 5 shows a subset of this DB for the first 25 cities and 17 tumour types (from the total number of classes).

The constrains have been simulated calculating a random figure (from 0 to 29) at every node for every dataset. '0' represents that there is no dataset available of that tumour type and no constrain can be assigned for delivering nothing. Any number between '1' and '28' indicates the reference of the tumour type to be delivered by the requesting node (as constrain for authorising the access to the data). '29' indicates that there is no constrain to fulfill and the cases available at the node for that specific tumour type will be freely delivered to the requesting node. Table 6 shows a subset of these constrains for the first 25 cities and 17 tumour types[1].

Due to the time constrains during the simulations of the protocol behaviour with different TTL values, a subset of the whole DB has been created. Considering that the CBTRUS data comes from USA, a DB including only the 205 cities from USA of the initial DB has been used for the simulations.

A.1 Simulation output

After the simulation execution a DB with the total number of cases collected per node and datatype is created.

Table 4 shows a subset of that database after the execution of the simulator with TTL=20. Its content corresponds to the following:

- **Node**: Requesting node
- **R**: Resource requested
- **C**: Initial number of cases of type R at the requesting node
- **MCP**: Number of MCP participating in the multilateral agreement
- **MSG**: Number of messages exchanged
- **CC**: Number of new cases of type R collected from the network
- **Path**: Average length of the multilateral agreement path

All the results presented in this article have been obtained after the processing of these figures.

[1]The complete DB used for the simulations from which the figures of tables 3 and 4 have been obtained is freely available upon request

Table 3: CBTRUS Statistical Report (2004-2007)

Type	Histology	Cases
A	**Tumors of Neuroepithelial Tissue**	**76340**
A1	Pilocytic astrocytoma	3663
A2	Protoplasmic and fibrillary astrocytoma	1242
A3	Anaplastic astrocytoma	4747
A4	Unique astrocytoma variants	1097
A5	Astrocytoma, NOS	5194
A6	Glioblastoma	37890
A7	Oligodendroglioma	3184
A8	Anaplastic oligodendroglioma	1386
A9	Ependymoma/anaplastic ependymoma	3011
A10	Ependymoma variants	1135
A11	Mixed glioma	2251
A12	Glioma malignant, NOS	4963
A13	Choroid plexus	511
A14	Neuroepithelial	236
A15	Non-malignant and malignant neuronal/glial	3169
A16	Pineal parenchymal	404
A17	Embryonal/primitive/medulloblastoma	2257
B	**Cranial and Spinal Nerves**	**19605**
B1	Nerve sheath	19600
C	**Tumors of Meninges**	**80457**
C1	Meningioma	77908
C2	Other mesenchymal	667
C3	Hemangioblastoma	1882
D	**Lymphomas and Hematopoietic Neoplasms**	**5380**
D1	Lymphoma	5380
E	**Germ Cell Tumors and Cysts**	**1092**
E1	Germ cell tumors, cysts and heterotopias	1092
F	**Tumors of Sellar Region**	**31405**
F1	Pituitary	29806
F2	Craniopharyngioma	1599
G	**Local Extensions from Regional Tumors**	**194**
G1	Chordoma/chondrosarcoma	194
H	**Unclassified Tumors**	**12318**
H1	Hemangioma	1814
H2	Neoplasm, unspecified	10373
H3	All other	131
	TOTAL	**226791**

Table 4: DB generated after the protocol execution.

Node	R	C	MCP	MSG	CC	Path
Akron	A1	0	2	20	10	3
Akron	A2	0	6	156	10	7
Akron	A3	10	2	20	70	3
Akron	A4	0	8	272	10	9
Akron	A5	10	7	210	10	8
Akron	A6	90	6	156	40	7
Akron	A7	0	6	156	10	7
Akron	A8	0	3	42	10	4
Akron	A9	0	2	20	10	3
Akron	A10	0	2	20	10	3
Akron	A11	0	1	6	10	2
Akron	A12	10	5	110	10	6
Akron	A13	0	1	6	30	2
Akron	A14	0	1	6	10	2
Akron	A15	0	1	6	10	2
Akron	A16	0	1	6	20	2
Akron	A17	0	4	72	10	5
...						

Table 5: Cases per node (city) and tumour type from the DB used for the simulations (sample)

City		Brain tumour type																
		A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17
1	Akron	0	0	10	0	10	90	0	0	0	0	0	10	0	0	0	0	0
2	Albany	10	0	10	0	10	120	10	0	0	0	0	10	0	0	10	0	0
3	Albuquerque	0	0	10	0	10	90	0	0	0	0	0	10	0	0	0	0	0
4	Allentown	0	0	10	0	10	80	0	0	0	0	0	10	0	0	0	0	0
5	Amarillo	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0
6	Anaheim	30	10	40	10	50	390	30	10	30	10	20	50	0	0	30	0	20
7	Anchorage	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0
8	Ann Arbor	0	0	10	0	10	70	0	0	0	0	0	10	0	0	0	0	0
9	Appleton	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0
10	Asheville	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0
11	Atlanta	30	10	40	10	50	370	30	10	20	10	20	40	0	0	30	0	20
12	Atlantic City	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0
13	Augusta	0	0	0	0	0	60	0	0	0	0	0	0	0	0	0	0	0
14	Austin	10	0	20	0	20	170	10	0	10	0	10	20	0	0	10	0	10
15	Bakersfield	0	0	10	0	10	90	0	0	0	0	0	10	0	0	0	0	0
16	Baltimore	20	0	30	0	30	280	20	10	20	0	10	30	0	0	20	0	10
17	Baton Rouge	0	0	10	0	10	80	0	0	0	0	0	10	0	0	0	0	0
18	Beaumont	0	0	0	0	0	50	0	0	0	0	0	0	0	0	0	0	0
19	Biloxi	0	0	0	0	0	50	0	0	0	0	0	0	0	0	0	0	0
20	Binghamton	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0
21	Birmingham	10	0	10	0	10	120	10	0	10	0	0	10	0	0	10	0	0
22	Bismarck	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0
23	Boise City	0	0	0	0	0	50	0	0	0	0	0	0	0	0	0	0	0
24	Boston	30	10	50	10	50	400	30	10	30	10	20	50	0	0	30	0	20
25	Boulder	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0
...																		

Table 6: Constrains per node and dataset (sample)

City		Brain tumour type																
		A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17
1	Akron	0	0	4	0	26	5	0	0	0	0	0	5	0	0	0	0	0
2	Albany	3	0	16	0	2	4	7	0	0	0	0	18	0	0	5	0	0
3	Albuquerque	0	0	25	0	1	25	0	0	0	0	0	2	0	0	0	0	0
4	Allentown	0	0	6	0	22	5	0	0	0	0	0	11	0	0	0	0	0
5	Amarillo	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
6	Anaheim	21	3	5	27	6	3	10	19	18	2	5	25	0	0	25	0	4
7	Anchorage	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0
8	Ann Arbor	0	0	27	0	11	9	0	0	0	0	0	10	0	0	0	0	0
9	Appleton	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0
10	Asheville	0	0	0	0	0	22	0	0	0	0	0	0	0	0	0	0	0
11	Atlanta	25	12	1	14	5	28	11	1	13	15	9	13	0	0	16	0	7
12	Atlantic City	0	0	0	0	0	25	0	0	0	0	0	0	0	0	0	0	0
13	Augusta	0	0	0	0	0	11	0	0	0	0	0	0	0	0	0	0	0
14	Austin	16	0	17	0	26	16	18	0	6	0	4	6	0	0	13	0	15
15	Bakersfield	0	0	2	0	9	26	0	0	0	0	0	23	0	0	0	0	0
16	Baltimore	11	0	20	0	14	28	1	9	15	0	20	10	0	0	18	0	6
17	Baton Rouge	0	0	19	0	7	3	0	0	0	0	0	19	0	0	0	0	0
18	Beaumont	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0
19	Biloxi	0	0	0	0	0	21	0	0	0	0	0	0	0	0	0	0	0
20	Binghamton	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0
21	Birmingham	15	0	4	0	6	1	24	0	8	0	0	19	0	0	15	0	0
22	Bismarck	0	0	0	0	0	28	0	0	0	0	0	0	0	0	0	0	0
23	Boise City	0	0	0	0	0	25	0	0	0	0	0	0	0	0	0	0	0
24	Boston	23	1	23	13	17	10	9	15	2	11	3	17	0	0	10	0	24
25	Boulder	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
...																		

ePCRN-IDEA2: An Agent-Based System for Large-Scale Clinical Trial Recruitment

Seyi Feyisetan
Department of Informatics
King's College London
London, UK
seyi.feyisetan@kcl.ac.uk

Gareth Tyson
Department of Informatics
King's College London
London, UK
gareth.tyson@kcl.ac.uk

Adel Taweel
Department of Informatics
King's College London
London, UK
adel.taweel@kcl.ac.uk

Maria Vargas-Vera
Department of Informatics
King's College London
London, UK
maria.vargas-vera@kcl.ac.uk

Tjeerd Van Staa
General Practice Research Database
London, UK
tjeerd.vanstaa@mhra.gsi.gov.uk

Brendan Delaney
Department of Primary Care and Public Health Sciences
King's College London
London, UK
brendan.delaney@kcl.ac.uk

ABSTRACT

Clinical trials are a key method of evaluating the efficacy of healthcare interventions prior to public release. To allow clinical trials to take place, however, it is necessary to recruit sufficient patients for participation. Unfortunately, such recruitment poses a significant challenge though. In this paper, we discuss a novel agent-based system designed to enable patient recruitment; our system, ePCRN-IDEA, has been implemented and is currently under deployment in the UK healthcare system. Through this deployment, however, we have found a number of challenges relating to scalability; consequently, this paper focusses on an extension called ePCRN-IDEA2 that addresses these problems. Specifically, we place agents on General Practitioners' (GP) machines to dynamically compute patient eligibility in real-time during consultations, thereby enabling GUI notifications and immediate recruitment. In ePCRN-IDEA, all agents attempted to compute patient eligibility over all trials, resulting in a huge burden for a large-scale deployment (e.g. 100,000 trials). Therefore, in eCPRN-IDEA2, we have embedded the necessary intelligence in agents to dynamically compute eligibility over the trials that are most likely to match the clinic's and patient's characteristics. Through simulations, we evaluate the approach to show that our decentralised trial selection algorithm can achieve comparable performance to a global knowledge benchmark with far greater scalability.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: [Miscellaneous]; I.2.11 [**Computing Methodologies**]: Distributed Artificial Intelligence—*Intelligent agents*; J.3 [**Life and Medical Sciences**]: Medical Information Systems

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), June, 4–8, 2012, Valencia, Spain.

General Terms

Design, Performance

Keywords

Clinical Trial Recruitment, Agent-Based Systems, Scalability

1. INTRODUCTION

Clinical trials are the gold standard by which medical research is evaluated. They are used to study various aspects of medical science, as well as being a vital stage in the deployment of new drug treatments. In essence, they involve the testing of new medical theories (e.g. treatments) on real patients to study the effects. For instance, before release, a new drug intended to mitigate the effects of arthritis must first be tested on patients suffering from arthritis to ensure both efficacy and safety. Clearly, however, to enable this, it is necessary to recruit sufficient patients to allow meaningful statical results to be derived. A patient sample size of 10, for example, is unlikely to offer sufficient evidence to persuade regulation boards that a new drug is appropriate for release.

When performing patient recruitment, it is necessary to locate patients who fulfil well defined eligibility criteria, e.g. age, gender, illness etc. This is because, generally, only a small proportion of the population will exhibit the necessary characteristics to make them eligible for participation. For example, testing an arthritis treatment on a healthy individual will clearly not be able to validate its effectiveness. Consequently, it is evident that the recruitment stage of a clinical trial is vital for ensuring that (i) sufficient patients are recruited to enable meaningful statistical results to be gained, and (ii) all patients fulfil the required eligibility criteria to generate accurate results. Unfortunately, however, achieving these two requirements is often highly challenging with recruitment taking up to 30% of the clinical time line, and only 15% of clinical trials finishing on schedule [12]. In fact, a review of the UK Medical Research Council found that only 31% of trials actually recruited to their planned

targets. Clearly, this creates significant overheads, which results in patient recruitment costing 30 - 40% of the entire clinical trial costs. Consequently, improving this process is of paramount importance for the future success of medical research.

The main challenge for patient recruitment is locating and establishing contact with eligible patients within sufficient time to allow them to participate. Therefore, recruitment often involves human recruiters actively visiting clinics in an attempt to locate eligible patients (e.g. by searching local records). Unfortunately, however, this is often complicated due to the geographically distributed nature of primary care clinics (i.e. eligible patients can be thinly spread across many clinics). As such, recruitment can be extremely slow and expensive, particularly for large-scale trials dealing with rare conditions.

To address the above problem, we propose replacing human recruiters with software agents that reside at local clinics. Each agent would maintain a local repository of information about active clinical trials. Whenever a patient enters a clinic for a consultation, the agent would inspect information entered about them in real-time to ascertain if they are eligible for any trials. Through this, instant notifications could be presented to the General Practitioner (GP) during a consultation to inform him/her of the patient's eligibility in real-time. Consequently, this would allow patients to be immediately recruited, negating the need for laborious effort on the part of the patient, clinical researcher or human recruiter.

We have realised the above concepts in an agent-based recruitment system called ePCRN-IDEA [21, 22], which has been implemented in two versions. The first version is under deployment in the UK healthcare system, whilst the second (ePCRN-IDEA2) is a prototype extension currently under evaluation. The second prototype, which is focussed on in this paper, has been developed to address a specific scalability challenge that we found from the first version. In this first implementation, every agent in every clinic downloaded every trial description from a central repository. This, however, resulted in significant processing and storage overheads because each agent was then required to compute the eligibility of a patient against every trial in the system; this could be a huge number, for instance, well over 100k trial are listed on clinicaltrials.gov [2]. Unfortunately, the limited resource capabilities of typical GP machines, as well as the complex nature of certain eligibility criteria, mean that such an approach is unscalable. Therefore, to address this, we have developed a second version, which allows agents to inspect and learn the characteristics of their host clinics to intelligently select the trials that they are most likely to find recruits for. This allows such agents to focus on processing the most suitable trials in a targeted manner. Further, through this intelligence, it becomes possible for agents to cluster into similar groups of trial interest, thereby allowing them to securely share their local repositories rather than using the central store. Through these extensions, we hope to move towards a far larger (pan-European) deployment of ePCRN-IDEA2.

This paper details and evaluates the components and algorithms used in ePCRN-IDEA2, with a focus on ensuring that the system can scale up with increasing numbers of clinics and trials. Specifically, our contributions are as follows:

- An assessment of traditional recruitment approaches and current Clinical Trial Alert systems highlighting their non-scalable nature.
- An extension and evaluation of the ePCRN-IDEA system to ensure scalability in the face of increasing numbers of trials.
- An extension and evaluation of the ePCRN-IDEA system to reduce the loading on a central server by allowing agents to cooperatively share trial information.

The rest of the paper is structured as follows: Section 2 gives the background to the research, leading to the design of ePCRN-IDEA2 in Section 3. Afterwards, an evaluation is then presented in Section 4, ending with the conclusion and future work in Section 5.

2. BACKGROUND

This section presents the background to the research. It first discusses clinical trial recruitment, before talking about scalability and, more generally, about agents in healthcare.

2.1 Clinical Trial Recruitment

Clinical trials are a challenging stage in the research of clinicians due to the complexity of recruiting patients for participation. Many types of trials can suffer from such difficulties; for instance, trials that have potential recruits who are widely distributed over many clinics (e.g. primary care) are extremely difficult to recruit for due to the intensive resource requirements. Studies show that 30% of participating clinics fail to even recruit a single patient [15]. Further, this can be exacerbated by many concerns, especially when dealing with complex eligibility requirements or trials that require immediate actions (e.g. a change of drug treatments).

Clinical trial recruitment is performed by first defining *eligibility criteria* that stipulates the exact characteristics that make a patient eligible for participation (e.g. gender, age, ailments etc.). Once this has taken place, it is then necessary to discover patients who match the criteria, before contacting and recruiting them. Traditionally, locating such patients is achieved using one or more of the following approaches:

- Advertising and public relations: This involves using posters, adverts and brochures to advertise eligibility criteria directly to practitioners and patients.
- Recruiters: This involves sending human recruiters to clinics, usually after feasibility modelling, analysis and site selections, in an attempt to discover patients who match the eligibility criteria.
- Practitioners: This involves doctors meeting periodically to discuss patient treatments and potential trials in an attempt to spot eligible patients during consultation.

These methods, however, are highly time consuming and expensive, particularly for trials that have high patient targets, complex eligibility criteria, rare diseases or involve emergency cases. This has led to the development of Clinical Trial Alert (CTA) systems, which alert practitioners to the eligibility of a patient when they are in consultation. Such systems then allow the practitioner to immediately discuss the trial with the patient, to enable instant recruitment in a

trusted environment (usually through a web interface). This process is achieved by automatically comparing patient information against computable eligibility criteria in real-time during consultations. However most of these systems [8, 4, 6] cater for recruiting patients to a single trial within a single clinic. Other similar techniques have also seen only limited large-scale testing [18]. The challenge of designing generic systems which can handle multiple trials, however, is hampered by the need to perform complex eligibility matching in real-time. Clearly, doing so for large numbers of diverse trials can make the process highly challenging in terms of performance. As of yet, this has led to simplistic CTA systems, which generally deal with small individual trials. We therefore believe that it is vital to address such challenges to enable the deployment of a generic scalable CTA system that can have a real impact on (global) clinical trial recruitment.

2.2 Scalability

Scalability can be defined as the ability of a system to operate within an acceptable performance range in the face of scaling up alternate system parameters (e.g. number of nodes). Currently, most systems adhere to some variation of the client-server model in which a single (logical) server handles requests issued by a number of subordinate clients (as opposed to hybrid and peer-to-peer models [20]). This, for instance, is how the above CTA systems operate, as well as the original ePCRN-IDEA implementation. Clearly, however, this does not scale as a centralised point can only possess a finite amount of resources, whilst the number of subordinate clients can continually increase with ease.

In the context of ePCRN-IDEA, there are two system parameters of importance for ensuring scalability: the number of agents and the number of trials.

As the number of *agents* (clients) increase, the loading on the server similarly increases; consequently, after a certain population is reached, the centralised resources must be upgraded to continue an acceptable quality of service.

Similarly, as the number of *trials* increase, the load on the agents also increases as it becomes necessary to compute a patient's eligibility over a larger set of eligibility criteria within a very strict time frame, i.e. before the patient has left the clinic.[1] This latter point is particularly difficult to manage because it is not possible to conveniently upgrade the resources of each agent as they are distributed throughout the entire country (there are approximately 10k clinics in the UK alone). Unsurprisingly, most GP clinics tend to utilise relatively low resource computers with limited storage capabilities, making it impossible to handle large numbers of trials (e.g. a single trial could be approx ≈1 MB). For instance, our measurements show that a typical desktop machine can take up to 100 ms to compute patient eligibility for a single trial; this means a trial repository size of 100k [2] could take over two hours to process per patient. Thus, it becomes necessary to conceive new ways to improve scalability without over-utilising or extending computing resources.

2.3 Agent Based Healthcare Systems

Agents have emerged as a prominent technology for handling a range of real-world problems [11]. Agents in healthcare have seen widespread investigation; Nealon et.al [14] discussed 11 areas in which agent technology is being applied to support and improve healthcare in Europe. Some of the areas discussed include using agents to integrate [13] heterogeneous patient records, using agents to control cardiac pacing and monitoring the elderly using agent-based teleassistance.

For example, MAID [7] is an agent-based system for integrating heterogeneous data sources within a hospital environment. The hospital studied had 24 departments, each using their own information systems. To address this, agents were constructed to interoperate with each system to monitor changes and retrieve data for insertion into a central repository. In a subsequent work, HealthAgents [9] went beyond MAID to also enable decision support, specifically for diagnosing brain tumours.

A range of agent-based systems have also been proposed for handling distributed expertise. These includes using agents to enable better communication between healthcare workers based on ambient information, e.g. their role, location etc. [17], as well as using agents to remotely monitor patients [10][16]. These systems also often involved data analysis; S(MA)^{2}D, for instance, uses statistical analysis to cluster patients into similar groups [16]. This ability to scalably perform data analysis in real-time, clearly, also shows potential for enabling the type of eligible patient identification discussed previously. Despite this, so far little work has been performed into using agents to improve clinical trial recruitment. Consequently, the rest of this paper explores exploiting the properties of agents to enable scalable patient recruitment.

3. EPCRN-IDEA2 SYSTEM DESIGN

This section presents the ePCRN-IDEA2 recruitment system, which is used to notify GPs of patients' eligibility during consultations.

3.1 Overview

The central aim of our research is to build a scalable system for clinical trial recruitment. At a high level, the system consists of two agents, as shown in Figure 1:

- Trial Agent: This resides at a central point. It holds a record of all available trials, which can then be accessed by GP Agents.
- GP Agent: This resides at a local clinic on a GP's machine. It retrieves trials from the Trial Agent and stores them locally. Whenever a patient enters a clinic, it compares his/her data with all known trials to compute if he/she is eligible for a clinical trial. If so, a pop-up is generated to notify the GP and to allow the patient's immediate recruitment through a web interface.

3.2 Trial Agent

The Trial Agent controls access to the central trial repository, holding an active record of all the trials in the system. It also manages request handling and trial transfers to the GP Agents. On startup, the Trial Agent loads all trials into its local repository. It then registers itself with a Yellow Pages service, which allows other agents to discover its services. In essence, the Trial Agent consists of two main components: the Trial Repository and the Trial Updater. We now briefly cover each of the Trial Agent's functions.

[1]On average, a consultation will last ≈10 minutes.

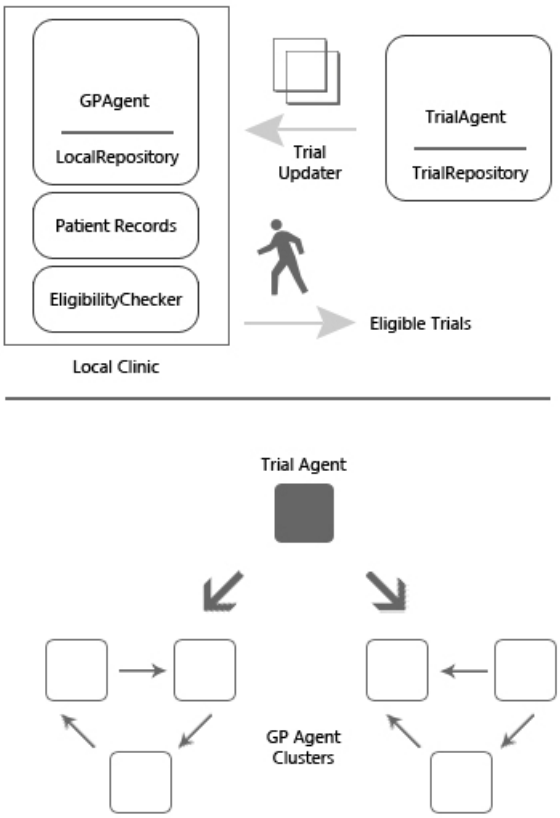

Figure 1: Agent System High Level Design

The Trial Repository component holds a description of each trial, represented using a standardised model e.g. the Biomedical Research Integrated Domain Group (BRIDG) [1] or the Primary Care Research Object Model (PCROM) [19]. These are standard information models that have been developed by clinicians to represent all relevant aspects of primary care research. Currently, PCROM is used, thereby requiring any clinical recruiters to define their trial information in this format. Put simply, PCROM is an XML schema that defines a number of attributes that must be used to describe each trial. To assist in its usage, a user interface has also been developed to automatically generate the format. In terms of ePCRN-IDEA2, the XML includes a trial description (e.g. name, brief overview etc.), a recruitment description (e.g. how many recruits are required) and the eligibility criteria (e.g. what characteristics must a patient fulfil to be considered eligible). Clearly, all clinical concepts in the system must be described using standard ontologies (e.g. Read Codes) to ensure semantic and syntactic interoperability. Collectively, these bodies of data offer the necessary information to decide if a patient is eligible and if the trial still requires recruits, before being able to present a description to the GP. Clearly, of most importance is the eligibility criteria, which can define rules for eligibility using a range of factors, including:

- Read Code(s): The patient must be associated with one or more Read Codes. Read Codes are standard codes that are used to describe clinical concepts, e.g. diagnoses, symptoms, social circumstances.
- Drug Code(s): The patient must currently be prescribed one or more drug treatments. Standard Multilex codes are used to represent drugs.
- Valid Patient List: The patient must be on a list of unique patient identifiers. These are usually generated at a central patient database (e.g. GPRD [23]), which contains collected patient records. This allows more sophisticated eligibility criteria to be pre-computed using full data sets and high performance resources. To ensure privacy, lists of patient identifiers are anonymised before being distributed using a one-way hash function. Mappings are then maintained by the organisation that generated the list of identifiers.

The Trial Agent also supports a variety of other types of criteria, as defined by PCROM. Importantly, combinations of these can be built to create more complex criteria. For instance, a typical form of eligibility criteria might include a list of potentially eligible patients plus a Read Code stipulating joint pain, i.e. to be eligible, one of the predetermined patients must enter the clinic and complain of having joint pain.

When a GP Agent wishes to retrieve trials from the Trial Agent, the request is processed by the Trial Updater component. This component is responsible for matching a GP Agent's characteristics (represented through certain parameters) to the available trials in the Trial Repository. The aim of this is to ensure each clinic retrieves the trials that they are most likely to be able to recruit on. The parameters currently consist of:

- A list of the registered patient identifiers in the clinic. This allows the Trial Agent to ensure that a trial using Valid Patient List eligibility criteria will only be sent to a clinic when the clinic contains one more eligible patients in the list.
- An ordered list of the top r most frequently observed Read Codes. This allows the Trial Agent to discern any specialisation in the clinic (e.g. cancer), to enable matching with eligibility criteria based on Read Codes.
- An ordered list of the top d most popular Drug Codes. This allows the Trial Agent to discern any tendencies to prescribe certain drugs in the clinic, to enable matching with eligibility criteria looking at particular drug usage.

These parameters therefore allow the Trial Updater to best match the clinic's characteristics to a bespoke subset of the globally stored trials. For example, a clinic that has a high number of cancer patients should receive trials that are mostly dealing with cancer. These parameters therefore determine what type of trials are forwarded to each GP Agent on an individual basis. Importantly, the parameters also determine how many trials should be sent based on the local repository size of the GP Agent. This then allows for variations in clinic resources, i.e. it allows clinics with higher capacity computers to locally process more trials. More formally, each request contains the following tuple $\langle V, R, D, n \rangle$,

where V: Set of clinic patients
R: Set of clinic top Read Codes
D: Set of clinic top Drug Codes
n: Clinic local repository size

We also define a function, $t = f(x)$ that retrieves a matching trial t based on an input search criteria x. A counter, i, also maintains the number of currently selected trials. A set, T, is then generated on each request containing all the trials to return to an agent, based on the previously discussed parameters; more formally,

T = Set of trials to be sent to the GP Agent

$$\forall v \in V : if(i < n) \longrightarrow t = f(v)$$
$$T = \{(t \notin T)\}i\text{++}$$
$$\forall r \in R : if(i < n) \longrightarrow t = f(r)$$
$$T = \{(t \notin T)\}i\text{++}$$
$$\forall d \in D : if(i < n) \longrightarrow t = f(d)$$
$$T = \{(t \notin T)\}i\text{++}$$
$$if(i < n) \longrightarrow t = f(rand(x))$$
$$T = \{(t \notin T)\}i\text{++}$$

In essence, the Trial Updater attempts to retrieve n trials by matching available trials with elements from the set V. Each successful match increments the counter i. This is repeated for the other parameters R and D until the size of set T is equal to n. Any extra slots are then filled up with randomly selected trials.

3.3 GP Agent

The GP Agent resides within the local clinic. It is responsible for sending request parameters to the Trial Agent to request relevant trials for its host clinic. It is also responsible for computing the eligibility of a patient in real-time during a consultation (and generating pop-ups). We now briefly cover each of the GP Agent's main functions.

3.3.1 Accessing Central Trial Information

First, to fill up its trial repository, the GP Agent runs an analysis of patients in its host clinic found within the patient records. These are stored in a local Electronic Healthcare Record (EHR) system; essentially, this is a database that is used to store information about each patient. Importantly, it is also actively used by GPs during consultations, thereby offering real-time information to the GP Agent. This allows the GP Agent to inspect information about a patient instantly, whilst the patient is still in consultation. The analysis on the EHR data results in a model detailing the most frequent diseases, most popular drugs and the list of patient identifiers in the clinic. These parameters are then used whenever the GP Agent requests new trials from the Trial Agent, as detailed above. Any retrieved trials are then stored in a persistent local repository to allow for fast access. Importantly, however, this local repository is of a finite size to ensure that the GP Agent is capable of both processing and storing the necessary trials. The default is 100, although this can be dynamically varied based on the memory, storage and processing capacity of the host. The above process is repeated every 12 hours to allow GP Agents to learn of any changes in the central repository.

3.3.2 Accessing Distributed Trial Information

The above section has detailed the default situation in which a GP Agent accesses trial information from the central Trial Agent. This, however, as previously mentioned, is not a scalable option as the number of GP Agents increase. This is because the central Trial Agent has only a finite amount of resources to service the GP Agents' requests. Thus, an increase in the number of GP Agents similarly requires an increase in the resources of the Trial Agent. Something which can be difficult in this domain due to the limited resources of academic research projects. Consequently, to address this concern, GP Agents are also allowed to access trial information from each other in an attempt to alleviate the burden on a central point (i.e. the Trial Agent). To achieve this, GP Agents cluster into groups of clinics that have similar characteristics, thereby allowing them to share trial information. This is because clinics with similar characteristics are likely to require similar trials, therefore allowing the decentralisation of trial distribution. Currently, the characteristics considered in this clustering are the most popular Read Codes and Drug Codes in the clinic. These clusters can be of any size based on the nature of the clinics being interconnected. Thus, to ensure security, all clinics must possess digital certificates, as well as only utilise encrypted communications.

Whenever a GP Agent starts up, it registers itself with the system's Yellow Pages service as a potential trial distributor (using the same service interface as the Trial Agent). Alongside this, it also registers its top illness and drug prescriptions in the clinic (accessed from the EHR software). Once it has done this, it then queries the Yellow Pages service to discover other GP Agents that have the same top diseases and/or drug prescriptions. If none are found, it simply utilises the central Trial Agent. However, if another GP Agent with the same top disease/drug prescription is discovered, it will simply clone its repository. When multiple are found the closest agent with the lowest loading is selected. The above process is then repeated periodically every 12 hours to ensure up-to-date trial information is maintained at each GP Agent.

Importantly, only trials based on Read and Drug codes are exchanged between the different GP Agents; this is for two reasons. First, any trials using Valid Patient List eligibility criteria will likely only be applicable to a small number of clinics (i.e. the clinics in which those patients are enrolled at). Consequently, there is (probabilistically) less benefit in sharing such trials. Second, sharing Valid Patient List eligibility criteria would likely raise certain concerns regarding patient privacy, as the inclusion of a patient on a Valid Patient List could potentially reveal a lot about that particular patient. Therefore, trials containing these lists are not shared.

3.3.3 Computing Eligibility and Recruitment

Once a GP Agent has a number of trials in its local repository, it can begin to compute eligibility for patients. This is performed in real-time whenever a patient enters the clinic. Specifically, the GP Agent is notified by the EHR system whenever a new consultation is opened. The EHR is used by the GP to enter and store information about patients, thereby offering a database of information to compute eligibility over. Importantly, any clinical information encoded in the trials (e.g. disease codes) must use the same syntax and semantics of the EHR data representation. Using this information, the GP Agent compares the patient data against the trials it is aware of to decide if the patient is eligible (e.g. are they the right age range, do they suffer from the correct illnesses etc.). If multiple eligible trials are found, a random one is simply selected; generally, patients will also only be recruited to one trial at a given time. Once

a match is found, the GP Agent generates a GUI pop-up to notify the practitioner, as shown in Figure 2. This pop-up allows the GP to register a response from the patient and to acquire extra information. Importantly, it also allows the patient to be immediately recruited through a web interface. In alternate situations, this web interface can also be used to recruit patients directly without using the GP Agent (e.g. if the GP independently decides a patient is eligible).

Figure 2: GP Agent Pop-up for Notifying Clinician of Patient Eligibility

4. EVALUATION

This section begins by taking a look at the methodology used to evaluate the system. We then seek to evaluate how effectively the system can scale up in terms of increasing numbers of trials and clinics. Specifically, we look at how effectively we can maintain performance and overheads in the face of these increasing variables.

4.1 Methodology

To evaluate ePCRN-IDEA2's scalability we have performed a number of system simulations. To achieve this, however, it is first important to create a realistic simulation environment and workload.

First, it is necessary to model how diseases are distributed throughout clinics (and patients). Second, it is necessary to understand what types of trials might typically be injected into ePCRN-IDEA2. Then, third, the characteristics and behaviour of the patients need to be modelled. Beyond this, it is also important to define the evaluative metrics that will be inspected. This section presents the evaluation methodology, looking at these four concerns. To achieve this, a prototype of the system has been built using the Java Agent Development (JADE) framework. This has then been used to perform a number of simulations with various parameter setups. An overview of the default setup is provided in Table 1; unless otherwise stipulated, these parameters are used in all experiments.

4.1.1 Modelling Disease Distribution

To present a realistic evaluation, it is important to have realistic data regarding disease frequency. This is so that the simulation can model the types of diseases that patients

Table 1: Default Parameter Setup

Parameter	Value
# Patients Per-Clinic	200
Total # Patients	5,080
# Consultations Per-Simulation	100
# Trials in Local Repository	100
Max # Trials in Global Repository	10,000
Distribution of Read/Drug Codes	Zipf
Skew of Read/Drug Codes (for Trials)	0.9
Skew of Read/Drug Codes (for Patients)	0.4
Total # Read/Drug Codes	10,000

are likely to report as suffering from. To achieve this, we use a Zipf Distribution [5]. According to Zipf's Law, the frequency of any disease is inversely proportional to its rank in the frequency table. This results in a small number of diseases being frequently encountered (e.g. flu) and a 'long tail' of diseases that are far rarer (e.g. papillitis). This can therefore be used to model the ailments reported by patients during consultations. To validate this choice, the generated distribution of diseases was compared against data provided by the World Health Organization (WHO) [3], which confirmed its accuracy. Diseases were generated using an alpha parameter (skew) of 1 and a set size of 10,000 to mirror the WHO distribution.

4.1.2 Trial Workload

Clearly, it is important to model realistic trial workloads in the system. To do this, we select a variety of possible types of eligibility criteria to test ePCRN-IDEA2 with. The trials generated were of four variants with eligibility criteria consisting of (*i*) Read Codes only, (*ii*) Valid Patient Lists only, (*iii*) Valid Patients and Read Codes, and (*iv*) Valid Patients, Read Codes and Drug Codes. These represent typical trial types that are usually encountered in clinical research. The rest of this section details how each trial type was generated. Each trial's eligibility criteria was generated with at least one of the following parameters:{V, R, D}, as described below:

Eligibility Criteria with Read Codes only (R): A random value r within a pre-set range $(1 - 2)$ is selected as the size of the set R. Read Codes are then assigned to fill up set R using the Zipf distribution (with a default skew of 0.9) from a global set of 10,000 Read Codes.

Eligibility Criteria with Valid Patient Lists only (V): A random value v within a pre-set range $(5 - 15)$ is selected as the size of the set V. Patient IDs are then randomly assigned to fill up set V from a set of 5,080 patient IDs. Clearly, a real clinical trial would use a far larger list size, however, scaling down both the pre-set range and global population size allows us to tractably emulate large-scale simulations.

Eligibility Criteria with Valid Patient Lists and Read Codes (V, R): This involved the two processes listed above to assign Read Codes and valid patients. Once the Read Codes and the valid patient list have been set, they

are combined and written into the trial.

Eligibility Criteria with Valid Patients, Read & Drug Codes (V, R, D): This involved the three processes listed above. A Drug Code is selected the same way the Read Code was selected. Once the coded information and the valid patient list have been created, they are combined and written into the trial.

4.1.3 Patient Workload

Last, it is necessary to simulate patients and their characteristics when arriving at clinics. The set of patients in a clinic is generated by random selecting patient IDs from the central patient database of 5,080 patients within the range assigned to the clinic. A patient history is then built for each patient by selecting a random number of visits between one and six, then assigning treatment Read Codes for each visit (using the Zipf distribution). This is then stored in the local EHR database of the clinic.

After creating the patient history, it is also necessary to simulate patient arrivals in the clinic (as eligibility is only ever checked when a patient is in consultation). To do this, on each simulation round, a random patient is selected for a visit. Read Codes and Drug Codes are then assigned for the current visit using a Zipf distribution (with a default skew of 0.4). The skew for each clinic is assigned based on clinic type, e.g. specialist clinics have a more skewed distribution.

4.1.4 Evaluation Metrics

Alongside the above parameters, it is also necessary to define the metrics by which we measure the scalability of the system. We do this through two values: performance and overhead.

We measure the *performance* of the system by the number of pop-ups[2] generated. Clearly, the ideal result is that every patient who enters a clinic and is eligible for a trial should be notified. However, practically speaking, this is not possible as it requires every GP Agent to know about every trial in the system (this is the original design of ePCRN-IDEA). Therefore, we use this global knowledge scenario as the benchmark by which we evaluate the effectiveness of ePCRN-IDEA2's approach of intelligently selecting trials on a per-clinic basis. Consequently, we represent the system performance as the percentage of pop-ups created in ePCRN-IDEA2 when compared against those that could have been generated if all agents knew of all trials.

We next measure the *overhead* of the GP Agent; in the above global knowledge benchmark, it is necessary to have very large local trial repository sizes, as well as massive processing capacities to compute eligibility in real-time. As previously mentioned, a global trial repository could take hours to process patient eligibility for, making the intelligent selection of trials vital. Consequently, to measure the overhead required to achieve a given performance level, we use the local repository size, as this is representative of not only the per-agent storage capacity required but also the processing costs for eligibility computation. Thus, we contrast the number of pop-ups a GP Agent can generate against the quantity of resources required to achieve them.

Last, we also measure the *overhead* of the Trial Agent, which is important when considering the feasibility of increasing the number of participating clinics in the recruitment system. To measure this, we simply use the number of active connections from GP Agents to the Trial Agent. This allows us to infer the loading that the Trial Agent has at any given time.

[2]A pop-up represents a patient who has been found eligible for an available trial.

4.2 Scaling the Number of Trials

As the number of trials increase in the system, it is important that ePCRN-IDEA2 can maintain a high number of pop-ups, whilst still ensuring each GP Agent does not get allocated too many trials to process. Ideally, GP agents will be able to keep a high number of pop-ups with only a limited size of local repository (i.e. high performance, low overhead).

To evaluate this, a number of different trial types (as described above) are tested in the system to measure their performance and overhead. Each trial type was evaluated using three different approaches to distributing trials from the central Trial Agent to the GP Agents. These are as follows: (*i*) distributing all trials to every GP Agent (global knowledge benchmark), (*ii*) retrieving a random set of n trial for each GP Agent, and (*iii*) intelligently selecting n trials based on the results of profiling the host clinic (i.e. using the algorithm presented in Section 3). The rest of the section presents results from simulating each trial type in ePCRN-IDEA2.

4.2.1 Trials with 1 Read Code (R)

The first type of trial tested simply contained eligibility criteria using a single Read Code (e.g. all patients who have diabetes). The single Read Code to be included within each trial was selected from a pool of 10,000 Read Codes using a Zipf distribution with a skew of 1. To evaluate the system, we compare the number of pop-ups with the theoretical maximum that would be possible by having all agents know about all trials.

Table 2: Trial Eligibility Criteria with 1 Read Code

All Trials		Random Trials		Selected Trials	
Trials	Pop-ups	Trials	Pop-ups	Trials	Pop-ups
1000	54	1000	4	1000	52
2000	65	2000	2	2000	52
3000	67	3000	1	3000	50
4000	70	4000	0	4000	51
5000	71	5000	0	5000	50
6000	72	6000	0	6000	51
7000	72	7000	0	7000	53
8000	73	8000	1	8000	53
9000	75	9000	0	9000	51
10000	77	10000	0	10000	50
Avg	70	Avg	1	Avg	50
Ovh	100%	Ovh	1%	Ovh	1%
Perf	100%	Perf	1.4%	Perf	71.4%

Table 2 details the results from the simulations. It can be seen that randomly selecting trials to fill up the local repository leads to a significantly lower number of pop-ups compared to the benchmark of global knowledge. This is due to the obvious difficulty of randomly selecting the most

appropriate Read Codes for a given clinic. In contrast, intelligently selecting trials with a particular Read Code based on the profile of the clinic led to a much higher number of pop-ups (an average of 50). In fact, for 10,000 trials, 65% of pop-ups could still be generated using only a local trial store of only 100. This is because the GP Agent was able to run a profile of its clinic and download trials that its patients were more likely to be eligible for. Importantly, performance remained relatively constant up to a size of 10,000 trials, without needing to extend the size of the local repository beyond 100.

4.2.2 Trials with Valid Patient Lists (V)

The second set of trials generated consisted of patients whose eligibility has been pre-computed at a centralised database. This form of eligibility criteria therefore simply consists of a list of all the patient identifiers who are eligible.

Table 3: Trials with Valid Patient Lists

All Trials		Random Trials		Selected Trials	
Trials	Pop-ups	Trials	Pop-ups	Trials	Pop-ups
1000	43	1000	4	1000	50
2000	72	2000	6	2000	52
3000	84	3000	6	3000	50
4000	89	4000	5	4000	54
5000	94	5000	7	5000	52
6000	98	6000	1	6000	53
7000	99	7000	4	7000	52
8000	96	8000	6	8000	51
9000	100	9000	2	9000	52
10000	100	10000	5	10000	50
Avg	87	Avg	5	Avg	52
Ovh	100%	Ovh	1%	Ovh	1%
Perf	100%	Perf	6%	Perf	60%

From Table 3, it can be seen that randomly filling the local repository led to only an average of only 5 pop-ups. This is because such trials can only recruit from a small number of clinics that the required patients are enrolled at. Consequently, random selections are highly suboptimal. In contrast, intelligently selecting trials based on the registered list of patients in the clinic resulted, on average, in 60% of the pop-ups of the global knowledge benchmark. This is because each GP Agent was able to fill its repository with most of the trials that its patients had been pre-computed as eligible for. This remained constant even when the local repository size was only 1% of the trial repository size. Consequently, even when the local repository size was increased to 200, intelligently selecting trials resulted in the same number of pop-ups as retrieving all the trials.

4.2.3 Trials with Valid Patients and 1 Read Code (V,R)

The third set of trial eligibility criteria generated consisted of a list of eligible patients who must also be diagnosed with a specific illness (e.g. Mr. Smith is eligible if he is also diagnosed with diabetes).

Intelligently selecting trials to fill up the local repository based on the patients in the clinic resulted, on average, in 75% of the pop-ups obtained from downloading all the trials. This was because the GP Agent was able to request trials based on its registered patient list and an analysis of its top Read Codes. This result remained constant even when the local repository size was only 1% of the central trial repository size. Even when the local repository size was increased to 200, intelligently selecting trials resulted in the same number of pop-ups as retrieving all the trials, indicating the scalability of the algorithm.

Table 4: Trials with Valid Patients and 1 Read Code

All Trials		Random Trials		Selected Trials	
Trials	Pop-ups	Trials	Pop-ups	Trials	Pop-ups
1000	4	1000	2	1000	7
2000	14	2000	2	2000	13
3000	17	3000	2	3000	18
4000	22	4000	4	4000	19
5000	23	5000	5	5000	22
6000	31	6000	2	6000	23
7000	30	7000	2	7000	24
8000	34	8000	2	8000	25
9000	34	9000	4	9000	24
10000	35	10000	2	10000	26
Avg	24	Avg	3	Avg	18
Ovh	100%	Ovh	1%	Ovh	1%
Perf	100%	Perf	12%	Perf	75%

4.3 Scaling the Number of Clinics

As the number of clinics (and GP Agents) increase, the loading on the central Trial Agent similarly increases. Consequently, to ensure scalability, we consider it necessary to better utilise the global system resources to alleviate this burden. In the context of ePCRN-IDEA2, the goal is to decentralise the distribution of trials as much as possible, thereby reducing the burden on the central Trial Agent. This involves GP Agents connecting to other GP Agents to request trials, rather than always utilising the Trial Agent. To measure this, we use the number of dependent connections to the central Trial Agent as an overhead metric. Clearly, this indicates the level of loading on the central repository and should therefore be kept low.

Figure 3 presents an exemplary graph of the GP Agent interconnections during a 10 node simulation. It can be seen that the agents are clustering together based on their clinic's characteristics. Importantly, only 2 nodes were required to directly connect to the Trial Agent. To extend these results, simulations were performed with agent populations of up to 100. In each case, only 2 connections were maintained to the Trial Agent, with each GP Agent being able to effectively utilise its peers' resources.

These overhead results can also be contrasted with the performance achieved (measured using the number of pop-ups). To achieve this, the central Trial Repository was set-up with 10,000 trials, each with eligibility criteria consisting of 1 Read Code. Each GP Agent then retrieved a set of trials by either connecting to the Trial Agent or a peer GP Agent. The average number of pop-ups for each cluster size are presented in Table 5. The results here were very similar to those in Table 2 for 10,000 trials (around 50 pop-ups). This evidences the fact that the system can maintain a similar level of performance whilst also alleviating the loading on the central Trial Agent.

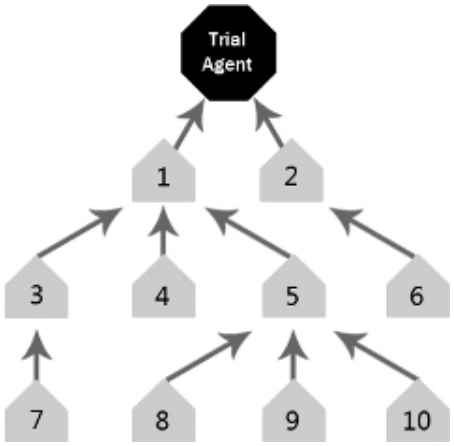

Figure 3: GP Agent Connections: Cluster of 10 GP Agents

Table 5: Trial Repository Size: 10,000 Local Repository Size: 100

All Trials		Selected Trials	
Cluster Size	Av. Pop-ups	Cluster Size	Av. Pop-ups
10	74	10	51
20	75	20	51
30	76	30	52
40	73	40	50
50	75	50	51
60	77	60	52
70	74	70	53
80	73	80	51
90	75	90	52
100	75	100	50

4.4 Summary and Discussion

Table 6 presents a summary of the scalability results; the percentages refer to the average performance and overhead levels of each selection method compared to the benchmark of global knowledge. As stated above, it can be seen that utilising random trial selections to address the global knowledge challenge resulted in consistently low performance, even though it does manage to maintain a low overhead (requiring only 1% of the global repository size). In contrast, it can be seen that ePCRN-IDEA2's approach of intelligently selecting trials results in significantly higher performance, whilst still maintaining very small local repositories. In fact, at its lowest performance, ePCRN-IDEA2 still manages to maintain 60% of the pop-ups that the global knowledge benchmark achieves.

Clearly, these results have therefore shown the unscalable nature of attempting to maintain global knowledge in a large-scale agent-based system. Our simulations have been based on repositories of 10,000 trials, however, this can easily extend well beyond this to in excess of 100,000 trials [2]. Consequently, ePCRN-IDEA2's approach is vital for ensuring scalable clinical trial recruitment. We have shown that it is possible to effectively target the distribution of trials on a per-clinic granularity. Specifically, in some settings, up to 75% of recruitment opportunities (pop-ups) can be maintained, even when reducing the local trial knowledge to just 1% of the global set. This suggests that large-scale recruitment can, indeed, be achieved without any need to upgrade local clinic computing resources. Beyond this, we have also evaluated the potential of reducing server loading at the Trial Agent by allowing GP Agents to share trial information. It has been shown that using peer GP Agents can easily reduce this load whilst, importantly, maintaining similar levels of performance.

Table 6: Summary of Performance and Overhead under Different Trial Types

	Random Trials		Selected Trials	
Trial Type	Perf	Ovh	Perf	Ovh
1 Read Code	1.4%	1%	71.4%	1%
Pat Lists	6%	1%	60%	1%
Pat List + Code	12%	1%	75%	1%

5. CONCLUSIONS AND FUTURE WORK

This paper has discussed the importance of clinical trials and the challenge of recruiting sufficient patients into them. It has looked at the current ways recruitment is carried out and the potential of using software agents to carry it out in a more scalable manner. This has led to the design and implementation of an agent-based system, ePCRN-IDEA2, which attempts to enable real-time patient recruitment on a large-scale. This system places agents in clinics with the responsibility of notifying practitioners whenever a patient, who is eligible for a clinical trial, is in consultation. This allows recruitment to be immediately performed before the patient has left the clinic. Further, to ensure long-term scalability, we have presented a way in which agents can intelligently select the trials that they consider themselves best able to recruit for (based on their host clinic). Through this, we have addressed the need for each agent to maintain global knowledge of all trials, thereby dramatically improving the ability of the system to scale up.

From this phase-2 prototype, we have identified a number of further lines of work. We believe it is important to extend the intelligence of the agents further, allowing them to gain a better understanding of their clinic. This, for instance, should involve improving the method by which agents can model a clinic's characteristics. This could also incorporate inferences regarding a given patient's likelihood to accept. Beyond this, it is clearly important to extend the evaluation to look at such things as larger trial bases, more divergent/convergent clinics and varying patient characteristics. Also, in this paper, the system has been evaluated under a synthetic workload; future evaluations should therefore endeavour to utilise more realistic setups. This will soon become possible using logging traces taken from the currently deployed ePCRN-IDEA system. Consequently, an important future step is using these traces to re-execute our simulations. Finally, it is our longer-term goal to also introduce this functionality into a working deployment, so that real results can be acquired regarding both performance and overheads.

6. REFERENCES

[1] Bridg project documentation and release notes for r3.

[2] Clinicaltrials.gov registry. http://clinicaltrials.gov/.

[3] Statistical annex 121. *The World Health Report 2004*, pages 2002–2004, 2004.

[4] L. B. Afrin, J. C. Oates, C. K. Boyd, and M. S. Daniels. Leveraging of open emr architecture for clinical trial accrual. *AMIA Annual Symposium proceedings AMIA Symposium AMIA Symposium*, 2003:16–20.

[5] S. K. Baek, S. Bernhardsson, and P. Minnhagen. Zipf's law unzipped. *New Journal of Physics*, 13(4):22, 2011.

[6] A. J. Butte, D. A. Weinstein, and I. S. Kohane. Enrolling patients into clinical trials faster using realtime recuiting. *Proceedings of the AMIA Symposium*, pages 111–115, 2000.

[7] R. Cruz-Correia, P. Vieira-Marques, P. Costa, A. Ferreira, E. Oliveira-Palhares, F. Araújo, and A. Costa-Pereira. Integration of hospital data using agent technologies - a case study. *AI Commun.*, 18, August 2005.

[8] P. J. Embi, A. Jain, J. Clark, S. Bizjack, R. Hornung, and C. M. Harris. Effect of a clinical trial alert system on physician participation in trial recruitment. *Archives of Internal Medicine*, 165(19):2272–2277, 2005.

[9] H. González-Vélez, M. Mier, M. Julià-Sapé, T. Arvanitis, J. García-Gómez, M. Robles, P. Lewis, S. Dasmahapatra, D. Dupplaw, A. Peet, C. Arús, B. Celda, S. Van Huffel, and M. Lluch-Ariet. Healthagents: distributed multi-agent brain tumor diagnosis and prognosis. *Applied Intelligence*, 30, 2009.

[10] V. Koutkias, I. Chouvarda, and N. Maglaveras. A multiagent system enhancing home-care health services for chronic disease management. *Information Technology in Biomedicine, IEEE Transactions on*, 9(4):528–537, dec. 2005.

[11] M. Luck, P. McBurney, and C. Preist. A manifesto for agent technology: Towards next generation computing. *Autonomous Agents and MultiAgent Systems*, 9(3):203–252, 2004.

[12] A. McDonald, R. Knight, M. Campbell, V. Entwistle, A. Grant, J. Cook, D. Elbourne, D. Francis, J. Garcia, I. Roberts, and C. Snowdon. What influences recruitment to randomised controlled trials? a review of trials funded by two uk funding agencies. *Trials*, 7(1):9, 2006.

[13] M. Nagy and M. Vargas-Vera. *Towards an Automatic Semantic Data Integration: Multi-agent Framework Approach.* Chapter in Sematic Web. In-Tech Education and Publishing KG, 2010.

[14] J. Nealon. Agents applied in health care. *AI Communications*, page 22, 2005.

[15] R. Nitkin. Patient recruitment strategies. Training workshop conducted by National Institutes of Health, Bethesda, Md, 2003.

[16] A. Rammal, S. Trouilhet, N. Singer, and J.-M. Pécatte. An adaptive system for home monitoring using a multiagent classification of patterns. *Int. J. Telemedicine Appl.*, 2008:3:1–3:8, January 2008.

[17] M. D. Rodríguez, J. Favela, A. Preciado, and A. Vizcaíno. Agent-based ambient intelligence for healthcare. *AI Commun.*, 18:201–216, August 2005.

[18] B. L. Rollman, G. S. Fischer, F. Zhu, and B. H. Belnap. Comparison of electronic physician prompts versus waitroom case-finding on clinical trial enrollment. *Journal of General Internal Medicine*, 23(4), 2008.

[19] S. M. Speedie, A. Taweel, I. Sim, T. N. Arvanitis, B. Delaney, and K. A. Peterson. The primary care research object model: A computable information model for practice-based primary care research. *Journal of the American Medical Informatics Association*, 15(5):661–670, 2008.

[20] G. Tyson, P. Grace, A. Mauthe, G. Blair, and S. Kaune. A reflective middleware to support peer-to-peer overlay adaptation. In *Distributed Applications and Interoperable Systems (DAIS)*. 2009.

[21] G. Tyson, A. Taweel, S. Miles, M. Luck, T. V. Staa, and B. Delaney. An agent-based approach to real-time patient identification for clinical trials. In *Proc. of the 4th Intl. Conference on eHealth (eHealth)*, 2011.

[22] G. Tyson, A. Taweel, S. Zschaler, T. V. Staa, and B. Delaney. A model-driven approach to interoperability and integration in systems of systems. In *Proc. of Workshop on Model-Based Software and Data Integration (MBSDI)*, 2011.

[23] T. Williams, T. Van Staa, S. Puri, and S. Eaton. Recent advances in the utility and use of the general practice research database as an example of a uk primary care data resource. *Therapeutic Advances in Drug Safety*, 2012.

An Agent Coordination Framework for IHE based Cross-Community Health Record Exchange

Visara Urovi
University of Applied Sciences Western Switzerland
Sierre, Switzerland
visara.urovi@hevs.ch

Alex C. Olivieri
University of Applied Sciences Western Switzerland
Sierre, Switzerland
alex.olivieri@students.hevs.ch

Stefano Bromuri
University of Applied Sciences Western Switzerland
Sierre, Switzerland
stefano.bromuri@hevs.ch

Nicoletta Fornara
Università della Svizzera italiana
Lugano, Switzerland
nicoletta.fornara@usi.ch

Michael I. Schumacher
University of Applied Sciences Western Switzerland
Sierre, Switzerland
michael.schumacher@hevs.ch

ABSTRACT

This paper presents an agent coordination framework that supports the exchange of Electronic Health Records (EHR) between health organisations using the profiles and standards proposed by the Integrating Healthcare Enterprise (IHE). We model a dynamic network of health organisations as connected communities that exchange and update health records of patients. The novelty of this work is twofold: it provides a general framework for coordinating EHR data exchange and it extends the current IHE profiles with the ability to dynamically connect to other communities that comply with such profiles.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Design, Experimentation

Keywords

Document exchange, EHR, IHE, Semantic Interoperability, Coordination, TuCSoN, OWL.

1. INTRODUCTION

Electronic Health Records (EHRs) refer to the systematic electronic collection of health information data about individual patients or populations [14]. The advantage of EHR over its paper-based version is that information can be quickly transfered, can support different views of the records and can be linked to best-practice guidelines to provide decision support [13]. Once EHR systems are in place, it is a natural step forward to seek integration of such records so that patient-centered access to health information is provided across institutional boundaries [27]. These requirements are becoming more urgent as the focus of health care delivery shifts from specialist centers to community settings [5].

The benefits of EHR integration are strongly dependent on the integration of the health care systems. Integrating the Healthcare Enterprise (IHE) is an initiative focusing on the integration of the healthcare information systems with the purpose to facilitate the exchange of patient information between healthcare professionals[1]. The IHE initiative (IHE) makes a major contribution to the integration of these systems and enjoys high acceptance due to its practical complement to existing standards such as HL7 CDA[2] [9] .

The IHE work is focused in specifying the integration of different clinical and organisational domains. For each domain, IHE maintains technical frameworks that contain all of the relevant information with regards to a specific domain. The most important part of the technical frameworks is the integration profiles. These profiles summarise domain specific use cases and communication scenarios based on standards. The significance of IHE stands on the fact that the profiles are constantly checked against practical experiences and are continuously adapted [27].

The provision of common technical frameworks for harmonising multiple-standards, causes IHE to have a high worldwide adoption with support from more than 400 member organisations[3]. Despite the significance of the IHE profiles, they lack features for handling dynamic scenarios where healthcare environments are dynamically connected to exchange data [18]. In such scenarios, mechanisms to discover other communities, the services and capabilities they expose and enable the cross-community data exchange are yet to be defined.

To address these problems, a system is needed where up-to-date patient's health records can be shared without prior knowledge of the health organisations that produced the data. In particular, semantic description of content has been recognised as a powerful tool for data sharing [15], that,

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), June, 4–8, 2012, Valencia, Spain.

[1] http://www.ihe.net
[2] http://www.hl7.org/
[3] http://www.ihe.net/governance/member_organizations.cfm

combined with agent-based computing, can contribute to automate the collection and processing of patient's EHRs. Agent-based systems can perform distributed communication and reason with semantic knowledge thus enabling EHR sharing between such heterogeneous systems. Furthermore, agent coordination models, such as tuple centres [24], that focus on decoupling the interaction amongst the actors, can contribute on making the different health actors more interoperable.

In this paper we propose an orthogonal solution to the existing IHE profiles that deal with EHR exchange. We propose an agent coordination framework that enables various health organisations to discover each other and to exchange EHR, whenever the organisational policies amongst the communities allow this to happen. The novelty of this work stands in the fact that we use semantics to automatically interpret the shared knowledge between different healthcare environments and to define the agent-based coordination mechanisms that coordinate the exchange and interpretation of such medical knowledge.

The rest of this paper is organised as follows: Section 2 describes our motivating case study, Section 3 introduces the IHE profiles; Section 4 introduces TuCSoN, a semantic-based framework for agent coordination; Section 5 describes how we engineer the agent-based coordination framework to deal with the exchange of EHR. We define the concept of community and define the coordination primitives that deal with event generation and notification; Section 6 describes parts of the implementation of the system; Section 7 evaluates the performamce of the system; Section 8 discusses relevant related work in the area of Semantic Interoperability and Multi-Agent Systems used for eHealth; finally, Section 9 concludes this paper summarizing its contribution and drawing the lines for future works.

2. MOTIVATING CASE STUDY

Our scenario is based in Switzerland, a federal country divided into 26 counties called cantons. The health system of Switzerland is a combination of public (i.e. hospitals) and private systems (i.e. doctors in private clinics) and health conditions can be treated in any of the competent healthcare providers. The Swiss Government has recently recommended the adoption of IHE profiles to achieve interoperability. The first pilot deployments have just been released, such as the eToile project [11] in Geneva.

In this scenario, Mrs Roux who lives in Lausanne, canton Vaud, is spending her holidays in Sierre, canton Valais. She suddenly needs urgent hospital care due to a strong chest pain. She explains to the receiving nurse that she had a heart surgery in the Hospital of Lausanne, which is also her home community and keeps all the updates of Mrs Roux health records. Such a community, does not necessarily has a copy of all the generated documents for Mrs Roux, but it knows where every document is stored. Mrs. Roux has also provided a privacy consent that establishes which data can be shared with other communitites.

Mrs Roux provides her insurance card to the nurse. Such card identifies Mrs Roux and it is used to search for her data in the Lausanne community. Based on the privacy concents given by Mrs. Roux, the query returns the meta-data information held on Mrs Roux (a list describing every document generated for Mrs Roux but not the documents themselves). The doctor who visits Mrs Roux is provided with the discovered information and can consult the documents of interest by retrieving the content from the community where the documents are stored. This is possible because Mrs Roux, through a web application, gave to medical doctors the right to access her medical data. Also, the rights to access Mrs Roux data can be overwritten in case of an emergency, provided that logs are created to monitor doctor's activities.

The doctor asks for further investigation tests to be carried out in the hospital of Sierre. After Mrs Roux' agreement, the tests together with the doctor's diagnosis are notified to the hospital of Lausanne. Under Mrs. Roux consent, also the general practitioner (GP) and the cardiologist curing Mrs. Roux are subscribed with the hospital of Lausanne to receive notifications of new generated data on Mrs Roux. Not only the hospital of Lausanne is now aware of this emergency case, and her new treatment, but also her two doctors.

After her return from vacation, the information has been already notified to the hospital of Lausanne, which in turn has notified it to the two interested private clinics where the two doctors work. Next time, when Mrs Roux visits such facilities, her doctor can view the relevant new information generated on Mrs Roux.

3. CURRENT IHE LIMITATIONS IN EHR EXCHANGE

The IHE Integration profiles are defined in terms of actors and transactions. Actors are components that act on information associated with clinical and operational activities in the enterprise. Transactions are interactions between actors that communicate the required information through standards-based messages. There are many IHE profiles that address interoperability between health care systems. We focus on those profiles that propose solutions for the exchange of EHRs.

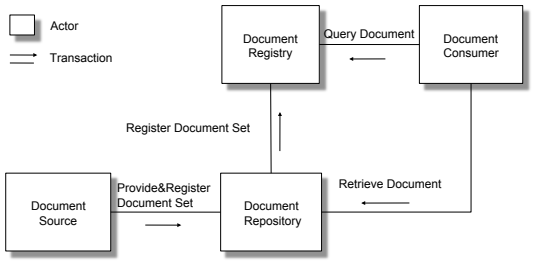

Figure 1: The XDS profile.

The Cross-Enterprise Document Sharing (XDS) [19] profile defines a coupling of health enterprises for the purpose of sharing patient-relevant documents. One or more healthcare enterprises that agree to work together under a XDS profile form an Affinity Domain [7]. The Affinity Domain can be viewed as a community specifying a set of policies, patient identifications and security mechanisms.

Figure 1 shows the XDS profile. At the core of XDS there is the document repository and document registry actors which respectively deal with storing health documents and storing meta-data about these documents to facilitate their discovery. The data are produced by a document source actor, typically a medical doctor in a hospital. A community may rely on more than one repository to store the produced

documents, however, all the meta-data must be stored and submitted within one registry. A document consumer actor can use the meta-data to know which repository contains the documents of interest. A patient identity source actor feeds patient identities to the registry.

Since XDS does not resolve document sharing among multiple affinity domains, the Cross-Community Access (XCA) profile specifies how medical data held by other communities can be queried and retrieved. XCA assumes that communities have pre-established agreements and knowledge of one another. It also assumes that the community which initiates a query towards another community, can determine the correct patient identifier of the patient under the authority of the receiving community [18]. This limits the operations in more dynamic scenarios where patients rely on the health documents to be available as they receive care in possibly different communities. The Cross-Community Patient Discovery (XCPD) profile is a newly proposed profile in trial implementation[4]. XCPD is defined to support the location of communities which hold patient's relevant health data and the translation of patient's identifiers across communities holding the same patient's data. The realisation of XCPD does not automatise the discovery of communities and it still requires communities to have pre-established agreements for exchanging the documents. In fact, the actor searching for documents in the cross-community must know beforehand which communities to contact.

If we were to model our case study only with the current IHE profiles, we encounter several limitations. We could use XCPD to locate Mrs Roux identity in the hospital of Laussanne. However, we need to assume that the Hospital of Sion has already agreements with the hospital of Lausanne to allow data exchange between the two. This is not often the case as nowadays patients move considerably and they may seek medical attention in different healthcare communities. Such communities may not necessarily know each other. This is particularly important in emergency cases such as the one described in our case study.

Even if every IHE community has pre-arranged agreements with every other IHE community, we still have the problem of propagating health information to the interested health providers. In fact, we can use the XCA profile to directly query the community of Lausanne about the data held on Mrs Roux, but, we have no way to define a proactive propagation of newly generated data. We can imagine that in a near future, patients will use different health services with some guarantee that their up to date health records can be viewed by the different service providers. The advantage of such integration is that it fosters better patient care and it helps to avoid mistakes that happen with limited patient information (i.e. allergies to specific medicaments in emergency cases).

In summary, the current practice is to identify external communities via manual exchange of configuration files. This approach is not effective in dynamic scenarios [18] that require health data to be promptly exchanged. Motivated by the lack of support mechanisms in the current IHE specifications, we define a complementary approach to enable communities to exchange data without necessarily having prior knowledge of each others. In this work we assume that a set of IHE compliant healthcare systems will be using the framework to discover and exchange information with other healthcare systems.

[4]`http://www.ihe.net/Technical_Framework/upload/IHE_ITI_Suppl_XCPD_Rev2-3_TI_2011-08_19.pdf`

4. SEMANTIC TUCSON

To address the issues presented in the previous section, we define an agent-based coordination model that supports communities to dynamically connect to one another and enables them to search, exchange and receive updates on patient relevant data.

Given that different communities may have different ways to present their information, we describe the data in terms of concepts and relationships amongst them. That is we specify an ontology that may be used to define the knowledge base of every community. This enables communities to interpret and reason on the data that are generated from different healthcare providers. In case two communities adopt different ontologies, mechanisms for ontologies reconciliation may be adopted.

Additionally, the IHE profiles define interactions as a simple message exchange. This means that in order to interact with other communities, it is imperative to know how to interface with them prior to any interaction. To overcome these limits the work presented in [2] uses Triple Space Computing (TSC) [10] as a general approach to combine semantics with coordination of messages for the purpose of EHR exchange. The TSC is based on the concept of blackboard systems, like Linda [12]. In TSC the interaction are mediated by shared tuple spaces. The tuple spaces are objects where interacting entities can write, consume and read tuples without necessarily having to synchronise (time decoupling), share the same space (space decoupling) or even know each other prior knowledge of one another (name decoupling) [12]. In addition to these advantages, the interacting entities communicate by writing and reading RDF triples making them schema decoupled too [2]. With these advantages, such mediation mechanisms improve considerably the interoperability of EHR exchanging systems as opposed to the simple message exchange.

The TuCSoN infrastructure defined in [21], takes the coordination aspects of TSC a step further. Apart from reading, writing and consuming semantic tuples, it allows to engineer additional primitives that coordinate the interacting entities.

4.1 Semantic Tuple Centres in TuCSoN

TuCSoN [24] is an agent coordination infrastructure based on tuple centres (entities where information is structured as tuples). Agents interact through tuple centres by inserting, reading and consuming tuples. Tuples are read and retrieved associatively. In order to read or retrieve a tuple, a tuple template has to be specified so that it can be used to find the requested tuple amongst all the existing tuples in the tuple centre [21]. The tuple centres can be syntactic, meaning that the structure of the tuple templates are known to the agents, or semantic, meaning that the information is produced and consumed following an ontology model.

Tuple centres are hosted in nodes and distributed in a network [4]. Each node can host as many tuple centres needed for the specific applications/systems. Additionally, the behaviour of the tuple centres is programmable by defining a set of coordination rules expressed in the ReSpecT language [23]. Using ReSpecT it is possible to define reactions that specify how a tuple centre reacts to incoming/outgoing com-

munication events. The reaction rules syntax is defined as follows:

reaction(action, guard, react).

where action is an operation made in a tuple centre (such as out(tuple)), guard discriminates between internal or external triggers and if the action is executed before or after the reaction takes place. For example, out(tuple) can be a communication event made by an agent or by a tuple centre. In guard it is possible to distinguish these two cases by respectively specifying if the incoming event is operational or internal. In guard it is also possible to state if the operation is executed in the tuple centre before the reaction is triggered or only after (respectively invocation or completion). Finally, react specifies a set of communication events that take place as a consequence of the performed action. The react part can include also some preconditions that should be verified in order for the communication events to be executed. In a ReSpecT reaction it is also possible to specify communication events (out, in, rd) towards other tuple centres. This allows to link different tuple centres with one another.

4.2 OWL DL and query language

TuSCoN uses the OWL Web Ontology Language [16] to model semantic tuple centres in terms of domain ontologies and objects [21]. Since 2004, OWL is a W3C recommended standard, it is a practical realization of a Description Logic known as $\mathcal{SHOIN}(\mathcal{D})$ [17]. Using OWL it is possible to define *classes* (also called *concepts* in the DL literature), *properties*, and *individuals*. An OWL ontology consists of a set of class axioms that specify logical relationships between classes, which constitutes a *TBox* (Terminological Box); a set of property axioms to specify logical relationships between properties, which constitutes a *RBox* (Role Box); and a collection of assertions that describe individuals, which constitutes an *ABox* (Assertional Box).

Classes are formal descriptions of sets of objects (taken from a non empty universe), and *individuals* are names of objects of the universe. Properties can be either *object properties*, which represent binary relations between objects of the universe, or *data properties*, which represent binary relationships between objects and data values (taken from XML Schema datatypes). Class axioms allow one to specify that subclass ($\sqsubseteq$) or equivalence ($\equiv$) relationships hold between certain classes and the domain and range of a property. Assertions allow one to specify that an individual belongs to a class: C(a) means that the object denoted by a belong to the class C; and that an individual is related to another individual through an object property: R(b,c) means the object denoted by b is related to the object denoted by c through the property R. *Complex classes* can be specified by using Boolean operations on classes: C $\sqcup$ D is the union of classes, C $\sqcap$ D is the intersection of classes, and $\neg$ C is the complement of class C. Classes can be specified also through *property restrictions*: $\exists$ R.C denotes the set of all objects that are related through property R to some objects belonging to class C at least one; if we want to specify to how many objects an object is related we should write: $\leq$nR, $\geq$nR, =nR where n is any natural number.

To realise the framework presented in this paper, we need to express some preconditions for the react part of the reaction rules. Every precondition can be a class assignment as defined by OWL DL, a query executed thanks to the reasoning services of a reasoning tool or, a Prolog predicate used to construct some specific function[5]. In order to execute a reaction, all its preconditions must be satisfied[6].

Given that there is not an official standard query formalism for OWL DL, in this paper we decide to adopt this one that is inspired from [3] and allows to express the queries that are available in the DL Query tab of Protègè[7]. In our implementation those queries are executed using the JENA API:

?-C $\sqsubseteq$ D $\Rightarrow$ true/false checks the subclass relationship;
?-C $\equiv$ D $\Rightarrow$ true/false checks class equivalence;
?-C $\Rightarrow$ true/false checks if the class is satisfiable;
?-C(a) $\Rightarrow$ true/false instance checking;
?-C(*) $\Rightarrow$ {a1,....an} retrieval, C can be a complex class.

5. AGENT-BASED COORDINATION FRAMEWORK FOR EHR EXCHANGE ACROSS COMMUNITIES

Our framework describes communities and their knowledge base in a semantic way and coordinates the interactions of those communities. Every community can be different from the others in terms of its organisation, the services it offers and the policies it uses. For this reason, we model the concept of community as an entity that exposes a set of services and its policies to enable interactions with other communities for the purpose of EHR exchange.

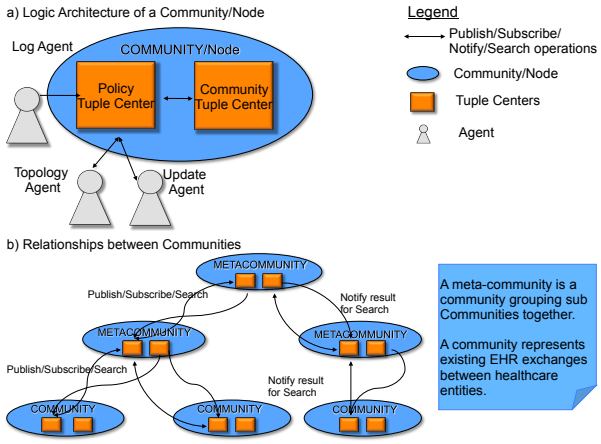

Figure 2: High view of the tree structured connections between communities.

We use a multi-agent platform to define the coordination mechanisms required for the communities to interact with one another. Fig.2(a) shows the architecture for one single community. The Policy Tuple Centre and the Community Tuple Centre define the coordination primitives in terms of

[5]In what follows we do not give the specification details of such predicates as they are intuitive, and, with a straightforward specification.
[6]As in Prolog, it is possible to specify preconditions to be executed in or by surrounding those with round brackets followed by a semicolon. Section 5.3 makes use of such specification.
[7]http://protege.stanford.edu/

action-reaction rules and are used to mediate interactions among the communities. The coordination primitives are coupled to a semantic tuple centre and are specified using the ReSpecT language[25].

At the moment we use a soft model of agency where agents simply react to specific messages exchanged in the tuple centres as opposed to a hard model of agency where the agents have complex cognitive models to perform complex reasoning. Nevertheless they are essential to keep the distribution and the automony of all the communities of the system. We delegate them specific tasks that are performed when specific events happen in a tuple centre. Thus, in every community, we specify three agents that are responsible for performing different actions. Fig.2(a) shows the Log Agent which is responsible for logging the different queries performed by other communities, the Topology Agent which is responsible for sending and capturing messages regarding the connection of a community with the rest of the structure and the Update Agent which is responsible for capturing updates and storing them into the knowledge base of the community.

Fig. 2(b) shows the communities organised in a tree structure. New communities are added to such structure in a self-organising manner. Each Node represents either an organisation with a whole community and the physical healthcare system behind it or can represent a meta-community representing a connection of communities in a more hierarchical structure. A meta-community can hold references to data that are physically stored by its children communities. The organisation of communities in a tree structure is useful when a community searches information for which it does not know the location. Since this type of queries tend to be computationally expensive, the organisation in a tree enables the propagation of queries to only those branches that have not been explored. This has the advantage that we do not broadcast an expensive query to all the known communities in the network which can potentially overload and multiply both the queries and the results. Furthermore, real world communities are usually organised following a tree structure due to their geographical disposition within a region and a state. Therefore, keeping a tree structure simplifies the representation of real communities within our system.

5.1 The Community Ontology

Every community has its own knowledge base. Other communities can query or subscribe to updates happening in more than one knowledge base.

Fig.3 shows the classes and the data and object properties of the OWL Community Ontology[8] that is used to create those knowledge bases. The classes are all disjoint. Due to space limitation and given that it is intuitive, we will not report here the range of all data properties. The *RBox* of the Community Ontology contains the following object properties (where the name of a property is followed by its *domain* and its *range*). The *TBox* contains the subsequent axioms that defines cardinality restrictions for the defined properties:

has : Patient → Document; InvFun(has);
cares : Community → Patient; InvFun(cares);
subscribe : Community → Patient;

[8]The full ontology can be found in http://aislab.hevs.ch/assets/OntologyCommunity.xml

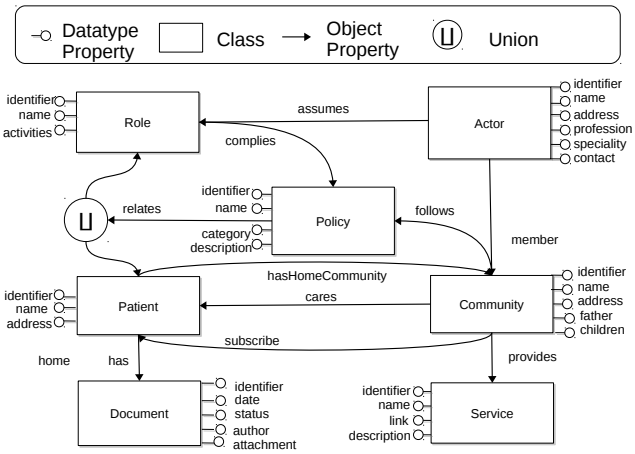

Figure 3: The OWL Community Ontology.

member : Actor → Community;
provides : Community → Service; InvFun(provides);
follows : Community → Policy; InvFun(follow);
assumes : Actor → Role; complies : Role → Policy;
relates : Policy → (Patient ⊔ Role); Fun(relates);
hasHomeCommunity : Patient → Community;
Fun(hasHomeCommunity);

Document ⊑ =1 has⁻; Service ⊑ =1 provides⁻;
Patient ⊑ =1 cares⁻; Actor ⊑ =1 member;
Policy ⊑ =1 follows; Policy ⊑ =1 complies;

The Community provides a set of Services, follows a set of Policies and cares about Patients. Each Community can subscribe to a Patient in another Community so that it is notified of the changes happening elsewhere. Each Patient of a Community has a set of Documents that are part of its health record.

Documents are generated and stored within a community. Every document relates to a specific patient. When a document is generated it has an author that is an actor in the community and a set of properties which indicates the content of the document. The community that generates such documents can also update their status by making documents obsolete or deleting them.

A Community has many Actors which can assume more than one Role. The actors are the users of the system, therefore they play roles such as a cardiologist, nurse, pharmacist, administration ect. The actors must act in the system by complying with the Policies of the Community. Such Policies define the actions that every role is allowed to perform. The Policies also help the integration of the Communities by making explicit under which rules the data are shared. In particular, they state which policies are applied to other communities that request patient data or subscribe to patient data in a community, or merge/delete themselves from the community structure. Such policies are enforced in the Policy Tuple Centre.

5.2 Policy Tuple Centre

The Policy Tuple Centre (PTC) deals with incoming re-

quests from other communities to either connect into a tree structure or to subscribe to notification of events. All the communications to a community are made to the PTC thus the Log agent is used to log all the interactions for future use. For every Community there can be a Father Community and many Child Communities (See Fig.3). PTC specifies the coordination primitives for adding, and removing a child or a father to a Community and the primitives for allowing subscription and unsubscriptions from other Communities.

5.2.1 Adding and Removing a Community

New communities can be added dynamically to the tree structure shown in Fig. 2. To achieve this we use the Topology Agent which executes the following steps:

1. It interfaces with a user to decide towards which community to perform a request or delete message and it writes the message in the PTC of the community that the user wants to connect/disconnect from;

2. It listens to accept, reject, add and remove messages generated in the Policy Tuple Centre;

3. In case it listens an accept message, it means that its community was successfully connected in the tree structure and it can add the father community to the knowledge base. It then goes back to listen the messages in step 2;

4. In case it listens an add message, it means that a new community was added to the tree structure. It adds the new community as a child community to the knowledge base and it goes back to listen the messages in step 2;

5. In case it listens a remove message it means that another community is requesting to be deleted from the tree structure, thus, it deletes the community from the knowledge base. If the removed community was the father, then it goes to step 1, otherwise it goes back to step 2;

6. In case it listens a reject message, it means that its community was not able to connect to the tree structure so the agent goes back to step 1.

The PTC specifies the following coordination primitive to a request message:

```
reaction(out(request(id, name, addr, policies),
  (operation, invocation),
  ?-Community(id) ⇒ false ,
  ?-Policy(*) ⇒ {p1...pn} ,
  subset(policies,{p1..pn}),
  out(add(id, name, addr),
  out(id, accept(myid, myname, myaddr) ).
```

The above reaction specifies that the community accepts as their children only those communities that are not already connected to the community and whose policies are a subset of its own policies. In a similar manner a community can be deleted from the tree structure. In order to delete a community, the Connect Agent sends a delete message to the father and, if any, to the children of the community that has to be deleted. The coordination primitive for such operation is defined as follows:

```
reaction(out(delete(id)),
  (operation,invocation),
  ?-Community(id) ⇒ true ,
  out(remove(id))).
```

The above primitive coordinates two Connect Agents. The first is the Connect Agent who is performing a delete action towards another community and the second is the Connect Agent of the community who is receiving the request. The second agent will remove the community from its knowledge base.

5.2.2 Subscribing to Community Events

A community can subscribe to events generated by other communities. We envisage three types of subscriptions: subscriptions to events regarding a patient, subscriptions to changes on the services a community offers and subscriptions to the changes of the policies that a community offers.

In this paper we treat only a simplified subscription mechanism for receiving patient updates from other communities. The two other subscriptions have similar considerations to the ones presented here. The Update Agent is used to subscribe its own community with other communities for the purpose of receiving patient updates. The Update Agent is involved when a health professional attempts to add/remove a patient into the knowledge base of the community, it executes the following steps:

1. If the home community and the community who is registering the patient differ, the agent generates a subscribe/unsubscribe message in the PTC of the home community of the patient. If the patient is added without a home community, then it generates a searchCommunity message. If the patient is added with partial data, then it generates a searchPatient message.

2. It listens to add, remove and reply messages in its own PTC;

3. In case it listens an add message, it means that a new community subscribes to specific events generated in its own community. The agent adds the new community and the subscribe relationship to the knowledge base and returns to step 1;

4. In case it listens a remove message, it means that a community is unsubscribing to specific events generated in its own community. The agent removes such community and its subscribe relationship from the knowledge base and returns to step 1;

5. In case it listens a reply message, it means that the resutls aswering a searchCommunity or a searchPatient query were found in the tree structure. The agent confirms the data with a human actor, adds them to the knowledge base and returns to step 1.

The coordination primitive for subscribing to patient updates is specified as follows:

```
reaction(out(subscribe(community, patient)),
  (operation, invocation),
  ?- Patient(patient) ⇒ true ,
  ?- Policy ⊓ (∃relates.{patient}) ⊓
  (∃category.{"filesharing"}) ⊓ (∃description.{"consent"})⇒ true,
  out(add(community, patient))).
```

The primitive is activated when another community requests a subscription to the PTC of a given `community` regarding the information of a given `patient`. In this case, the PTC checks that the identified patient is already contained in the knowledge base and that it exists a policy describing the patient consent into sharing its own files (the complex DL class is satisfiable).

When a new document regarding a patient is generated anywhere in the tree structure, the home community of the patient is notified by default. If the document is generated in the home community or an update about a patient arrives in the home community, such update is propagated to all the interested subscribers. The following coordination primitives deal with such a change:

Updates for a patient with a different home community

```
reaction(out(update(patient, document)),
  (operation, invocation),
  ?- Document(document) ⇒ true ,
  ?- Patient(patient) ⇒ true ,
  (∃homeCommunity⁻.{patient})(*) ⇒ {home},
  home ≠ myid,
  out(home,update(patient, document))).
```

Updates for a patient within the home community

```
reaction(out(update(patient, document)),
  (operation, invocation),
  ?- Document(document) ⇒ true ,
  ?- Patient(patient) ⇒ true ,
  (∃homeCommunity⁻.{patient})(*) ⇒ {home},
  home = myid,
  ?- Community ⊓ (∃subscribes.{patient})(*) ⇒ {c1...cn},
  out({c1...cn},update(patient, document))).
```

The above coordination primitives respectively specify that an update for a patient with a different home community from the one who generated the update must be propagated to the home community and if an update regarding a patient is generated in the home community all the subscribers to such an event should be notified. No agents are used in this operation as the PTC can directly update other PTCs by using TuCSoN coordination primitives.

In a similar way, communities may choose to unsubscribe to communities. This may happen because the patient is not anymore on care of the community and such updates are no longer necessary.

```
reaction(out(unsubscribe(community, patient)),
  (operation, invocation),
  ?- (∃subscribes.{patient})(community) ⇒ true,
  out(remove(community, patient))).
```

The above primitive specifies what happens in case of an unsubscribe request performed by an Update Agent. The PTC recieving the request checks first that the unsubscribing community has previously subscribed to the patient, then it generates a remove message that is captured by the Update Agent and which triggers the removal of the subscribe relationship between the requesting community and the specified patient.

5.3 Community Tuple Centre

The Community Tuple Centre (CTC) is an additional coordination module used to evaluate search queries that are generated in the system. Communities generate queries to search a new community or to search data related to a patient. Such search queries are computationally expensive thereby we evaluate them outside the PTC. In fact the PTC receives also requests for search queries, but it forwards them to the CTC which may either find the result of a query or propagate it to the PTC-s of the father and the child communities. In this way we evaluate expensive queries in parallel to the normal functionalities offered in PTC, and do not directly expose the CTC to the whole system.

5.3.1 Searching a Community or a Patient

A community can search other communities and patients by generating a query to the father community and to the children communities. If the community receiving the query does not find the requested data, it will propagate the query to its father and to its children if any (excluding the community that sent the query). The Update Agent is in charge of generating `communitySearch` or `patientSearch` queries. In the query message it indicates the sender, the community that is requesting the data and a list of criterias to be used for the search. The same query is propagated in the tree structure until the data is found following a flooding-like algorithm that stops when all the nodes are visited, for this reason, the sender changes as the query propagates.

In the search of a community or a patient some criteria may not be specified. For example, in an emergency case, the patient may not be able to produce a home community therefore the `homeCommunity` of the patient may be unknown. A community can still search the data of the patient by specifying some of the patient's demographic data. The coordination primitive for searching a patient is defined as follows:

```
reaction(out(patientSearch(community, sender,criterias)),
  (operation,invocation),
  Criteria1 ≡ Criteria2 ≡ Criteria3
  ≡ Criteria4 ≡ Criteria5 ≡ ⊤
  (member(('identifier', id), criterias),
    Criteria1 ≡ (∃identifier.{id}));
  (member(('name', name), criterias),
    Criteria2 ≡ (∃name.{name}));
  (member(('address', addr), criterias),
    Criteria3 ≡ (∃address.{addr}));
  (member(('community', community), criterias),
    Criteria4 ≡ (∃community.{community}));
  (member(('homeCommunity', home), criterias),
    Criteria5 ≡ (∃homeCommunity.{home}));
  ?-(Patient ⊓ Criteria1⊓ Criteria2 ⊓ Criteria3
   ⊓ Criteria4 ⊓ Criteria5)(*) ⇒ {p1,...pn }
  (empty({p1,...pn }, false),
    out(community, reply(myID, {p1,....pn})));
  (empty({p1,...pn }, true),
     ?- (Community⊓(∃ father.{"true"})
      ⊔(∃ child.{"true"}))(*) ⇒ {c1,....cn},
     remove(sender,{c1,....cn}, {c1,....cj}),
     out({c1,....cj},
         patientSearch(myID, community, criterias)));).
```

The above primitive specifies a class for the retrieve operation on the basis of the submitted parameters listed in the `criterias`. To test which are the specified criteria we use `member/2` Prolog clause. The primitive covers two cases: in the first case the list of specified criteria identify one or more patients that satisfy the query, in the second case no data are found that satisfy the query . The `empty/2` predicate discriminates the two cases. If `empty/2` predicate determines that there is one or more individual that has been found in the knowledge base, the result is sent to the community who was searching the data. If the `empty/2` predicate determines

that there are no individuals satisfying the query then the query is propagated to the father and the child communities. Before the propagation, we want to avoid the propagation of the query to the node that sent it. We specify the `remove/2` predicate to exclude the community `sender` that was sending the query from the list of communities. The specification of the coordination primitive for the communitySearch query is done in a similar way.

6. IMPLEMENTATION

The implementation of our framework is based on the TuCSoN semantic tuple centres as defined in [21]. Additionally, we interface with openXDS [9], an open source implementation of the XDS profile, in order to have documents stored and retrieved in an IHE compatible manner. Both of these infrastructures are JAVA based.

Figure 4 shows how the architecture of the system is related to TuCSoN. Every community is represented with a TuCSoN node and has its own semantic knowledge base. The semantic knowledge base is used by Jena to manage the defined ontology and from Pellet/SPARQL to perform queries and reasoning over the ontology. In order to provide the persistence of the data model, the semantic knowledge base is stored in a PostgreSQL database.

The User Agent is an external agent which interfaces with the users of the system. After a user's request to add information to the knowledge base takes place, the system generates the correct OWL assertions so that the new information can be added to the knowledge base. In case of the addition or modifications of documents, IHE compatible meta-data are generated to be stored in the registry of the XDS profile and the same meta-data are stored as semantic data in the community knowledge base. Updates to the meta-data of the documents of a patient are propagated towards the home community of the patient and to the subscribed communities. We assume that the actual fetching of the documents is realised using one of the existing IHE profiles (XCA already addresses this issue).

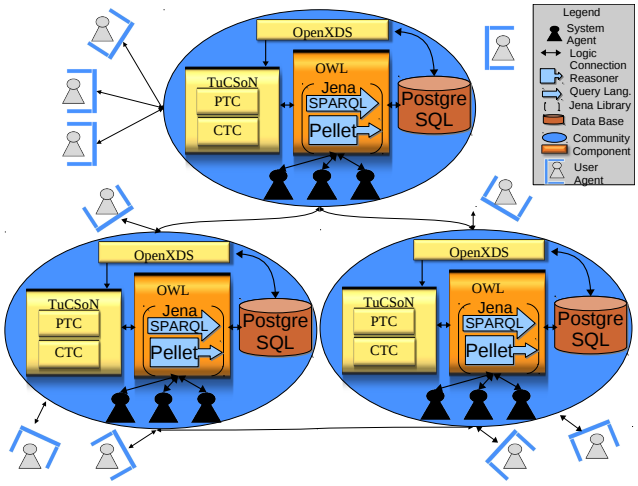

Figure 4: The implementation of the system.

[9] `https://www.projects.openhealthtools.org/sf/projects/openxds/`

There is a Topology Agent for each community node. The agent reacts to tuples generated by external User Agents or to tuples generated by the PTC of the community. The Topology Agent uses non-blocking in operations in the PTC and reacts to messages by either writing in the semantic knowledge base or by performing out operations in the PTC. The implementation of the Log and Update agent is realised in a similar manner. The behaviour of the topology agent in case of a request to connect to another community can be summarised as follows:

1. The User Agent requests the Topology Agent to connect to another community by performing an in operation of such request in the PTC of the community node. The Topology Agent reacts by creating an out operation in the PTC of the community identified by the User Agent. Such operation specifies a request tuple.
2. The PTC of the contacted community uses the ReSpecT reactions to determine if to accept or to reject the requesting community as its child (see first reaction of section 5.2.1). If the reaction is successful it generates an add tuple for its own Topology Agent and an accept tuple is inserted in the PTC of the requesting community. If this reaction fails, another reaction makes sure that a reject tuple is sent to the PTC of the requesting community.
3. If the add tuple is generated by the PTC, the Topology Agent consumes it and adds to the knowledge base the requesting community as a child
4. If the Topology Agent of the requesting community can consume an accept tuple in the PTC it adds to the knowledge base the new community as its own father.

The reaction primitives are specified by calling JAVA code from reactions specified in ReSpecT. This is possible because TuCSoN is based on tuProlog [6], a Java based implementation of Prolog that allows a seamless integration between Java code and Prolog predicates. For example, the coordination primitive to subscribe a community to patient updates is implemented as follows:

```
reaction(out(subscribe(Community, Patient)),
  (operation,invocation),
  in(subscribe(Community, Patient)),
  get_semanticKB(KB),
  KB←getBase returns Base,
  KB←getModel returns Model,
  java_object('coordination.UpdateUtility',
              [Model,Base],MyUpdateUtility),
  MyUpdateUtility←utilitySubscribe(Patient),
  out(updateAgent(add, Community, Patient)))).
```

The above reaction rule specifies that when an out of a subscribe tuple is made into the tuple centre, then the reference to the JAVA object representing the semantic knowledge base KB is used (the ← notation represent a call to a java module) to obtain the URI and the model Model of the ontology. We use an UpdateUtility java module to check if the policies allow us to subscribe the community Community to the patient Patient. MyUpdateUtility is a variable containing an UpdateUtility object constructed with the model Model and URI of the ontology. Finally, the tuple add is sent to the Update Agent which inserts the new information in the knowledge base.

7. EVALUATION

We performed tests to evaluate the proposed solution on a 24 cores Intel, 2.93 GHz and 96GB RAM. We defined 7 communities that we deployed on separate virtual machines and to which we assigned 1000 patients. Furthermore, for both of the tests, we gave to the 7 communities knowledge about other 1000 communities, which were not deployed in virtual machines. We did this to understand how the computation time of the different queries increases as the semantic knowledge base increases in size. In the first test we varied the number of subscriptions for a patient from 1 to 6 and measured the average time that a community takes to send updates regarding that patient to the subscribed communities.

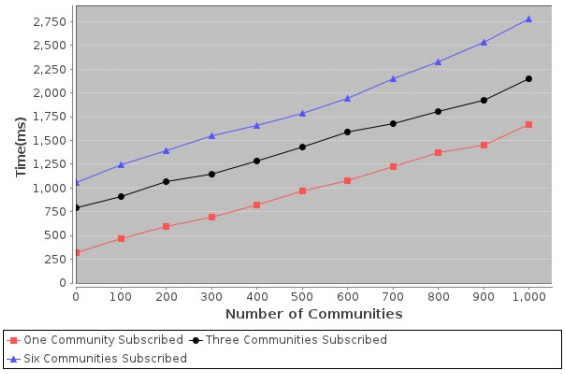

Figure 5: The Update Time.

Fig. 5 shows how the update time changes with a growing number of known communities and subscriptions to a patient. The time to update other communities grows linearly with the number of community individuals held in the knowledge base. This is due to the increase on the time to search for the communities that are subscribed to a specific patient. Also a growing number of subscriptions per patient introduces a latency as more than one update message has to be sent into other tuple centres.

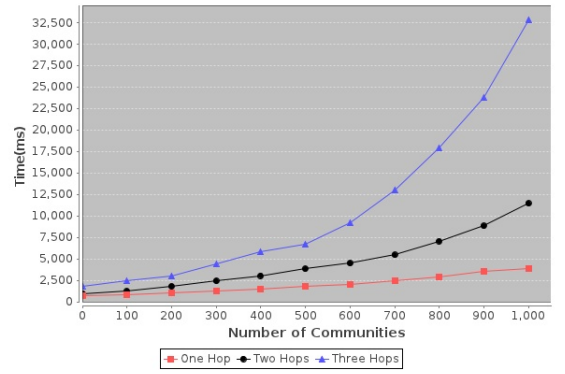

Figure 6: The time to search in other communities.

Secondly, we measured the time that a distributed search takes to find the results. We searched data that were at 1 then 2 and then 3 hops in the tree structure. Fig. 6 shows how the search time in other communities of the tree structure changes as the number of community individuals held in every knowledge base grows. The results show that the time to search in other communities grows exponentially as the search gets propagated in the tree structure. This is to be expected as the search has no prior knowledge as to where to send the query and each node has to evaluate the query before establishing that there is no data held and forward it to the neighbouring nodes. In our future work we will consider ways to partition and index data in an efficient way. The results are encouraging as it takes a few seconds to find the data while, in the current state of the art, in order to have interacting communities it takes several days of human intervention.

8. RELATED WORK

The use of semantic representations for the purposes of interoperability between hospitals is not a new idea [1, 8], nor it is new the idea to use the publish and subscribe pattern to model the dissemination of events in healthcare [26], but, to the best of our knowledge, the use of an agent-based coordination infrastructure to govern the semantic interoperability between distributed nodes representing communities is new. In particular, the epSOS project[10] aims at creating an integration broker for cross border exchange of patient's health records. In epSOS there is no mechanism defined to handle the subscription of new communities, thereby the responsibility to connect different healthcare providers falls into the epSOS operator. On the contrary, we propose the use of coordination primitives and agents technology to dynamically connect communities and have a more flexible approach towards subscription and notification of relevant events for a community.

The MediCoordination Healthcare Infrastructure (MHI) [1] aims at improving the accessing and sharing of important medical data between medical actors. MHI's architecture consists of a registry/repository and two clients, one for submitting documents and one for receiving them. An XDS-based server is used for the repository and the registry. The IHE XDS Integration Profile describes an infrastructure based on standards (ebXML), for managing the information exchange of sensitive medical data. The MHI prototype does not implement notifications [1]. General practitioners have to manually query the registry and client-server communications are channeled through a SOA-based service. With respect to MHI, we propose a solution that is decentralized and that can handle multiple communities, whereas MHI is limited to a centralized repository. Furthermore, our infrastructure makes use of Description Logic formalisms, allowing us a richer description of the events happening between different actors, across communities, and it also allows us to represent both subscription and notification to complex events.

As reported in [8], ARTEMIS is a project that provides interoperability for healthcare by semantically enriching Web services. Such semantically enriched web services are then connected by means of the JXTA P2P infrastructure. A JXTA super peer represents a community where different health actors can interact. The ontologies in the medical domain are mapped directly to different ontologies related to the heterogenous IT systems making use of ARTEMIS.

[10] `http://www.epsos.eu`

A set of mediator components is then used to consent heterogeneous services to communicate. We can say that our metacommunities and community entities, can be seen as the super peers of ARTEMIS. Nevertheless, our approach is quite different from the one proposed by ARTEMIS as we do not map the health concepts to a set of ontologies. This is because the problem of representing the health concepts is already addressed by standards like SNOMED[11]. We rather focus on enriching with semantics the existing documents to foster the interaction between existing communities, assuming that the health infrastructure of the health actors connecting to our system, will be capable to handle documents in HL7 format. This assumption is realistic as the adoption of HL7 based infrastructures is becoming a requirement for the existing health actors, as there is a necessity to foster communication across borders of regions and nations.

The Provenance project [20] defines a multi-agent system that records the patient's complete health care history (located in different communities). Each community creates and processes the documentation using p-assertions that are stored in a Provenance store. The p-assertions are assertions triggered by the documentation of a patient's health treatment process and indicate information such as which is the medical doctor who generated a record, what were the basis for the given treatment, and when the record was created. The metadata stored and exchanged in our system could be regarded as simple p-assertions because they are not meant to link the process that created such records. In our approach, the home community of a patient has a list of all the metadata created for each of the patient's health records. Therefore, similarly to the Provenance project, it is possible to query the documents by using the links that the metadata define to the actual documents. The documents are stored in the communities that created them and, in order to consult them, the users of a different community must undergo to the original systems' privacy protection. However, differently from the Provenance project, we do not enforce an additional description of the records into some other format, nor all the communities must interact with a central store to share the medical history of the patients. In our solution, the medical history of a patient can be found in its home community. The advantage here is that no prior connection with the home community of a patient is enforced (specially useful in emergency scenarios). Communities may join or leave the system and, based on the policies that apply to the record exchange, may or not allow subscriptions to a patient's health records. Whenever subscriptions are allowed, any new created metadata is automatically notified to other subscribed communities. Thus, there is no need to reconstruct the full medical history at every interaction with the patient. Also, the use of semantic data helps us to dynamically discover new communities and has the additional advantage that data which are represented differently in other communities, can still be interpreted in a meaningful way.

Triple space computing (TSC) applied to healthcare [22] is the approach that it is closer to ours. Also TSC uses tuple spaces to foster the exchange of information and proposes the use of semantic web technology to represent the data about the patients. Their solution associates RDF tuples to concepts defined in HL7 or SNOMED. This solution differs from ours as we are not concerned with translating HL7 concepts into a semantic web language but we deal only with the metadata associated to medical documents. From the perspective of the computation, also TSC considers the problem of publication and retrieval of health information, but it does not describe the notification and dispatching of the events happening in the distributed systems, nor there is a clear representation of the concept of community. Finally, by using tuple centres and ReSpecT, we can modify easily the behaviour of our communities, including new reactions at runtime, while this is not the case for TSC.

[11] `http://www.nlm.nih.gov/research/umls/Snomed/snomed_main.html`

9. CONCLUSIONS AND FUTURE WORK

In this paper we have presented a tuple centre based coordination model to enable dynamic cross-community interactions. We have shown how the combination of semantic representations and coordination languages such as ReSpecT can improve the current state of the art with respect to cross-community EHR exchange. The presented coordination model extends the current IHE limitations by specifying a set of coordination primitives to dynamically connect various communities in a tree structure.

Within such a model we enabled communities to search for new information in the distributed network of communities. Cross-community interactions were also generated in a proactive manner by allowing the communities to publish and subscribe to updates generated in other communities.

As part of our future work, we plan to address security issues arising from an open environment. Apart from the logging of events, the set of policies may help to check that the emerging behaviour of the actors performing the queries is that expected within the community sub-system. We will further investigate how to log the access to the data in a distributed setting in such a way that it is possible to track back all the access to documents. To this extend we also need to allow for secure authentication by exchanging of credentials and by connecting the communities and metacommunities to Certification Authorities.

We also plan to model subscriptions to different types of events (other than patient updates) and enable communities to apply filters to the exchanged information. Both of these extensions will require more complex semantic reasoning than the one presented in this paper.

10. ACKNOWLEDGMENTS

We would like to thank Bruno Alves and Marco Colombetti for their useful suggestions on the presentation of this paper. The work was partially funded by the Switzerland FNS grant nr. 200021 135386 / 1, the FP7 287841 COMMODITY12 project, the Hasler Foundation project nr. 11115-KG and by the SER project nr. C08.0114 within the COST Action IC0801 Agreement Technologies.

11. REFERENCES

[1] B. Alves, H. Muller, M. Schumacher, D. Godel, and O. Abu Khaled. Interoperability prototype between hospitals and general practitioners in Switzerland. *Stud Health Technol Inform*, 160:366–370, 2010.

[2] D. Cerizza, E. D. Valle, D. Foxvog, R. Krummenacher, M. Murth, and C. P. D. Milano. Towards European Patient Summaries based on Triple Space Computing, 2006.

[3] M. Colombetti. Dispense corso di Ingegneria della conoscenza: modelli semantici, Facolta di ingegneria dell informazione, Politecnico di Milano. http://home.dei.polimi.it//colombet/IC/materiale/IC2011.

[4] M. Cremonini, A. Omicini, and F. Zambonelli. Multi-agent Systems on the Internet: Extending the scope of coordination towards Security and Topology. In *Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World: MultiAgent System Engineering*, MAAMAW '99, pages 77–88, London, UK, 1999. Springer-Verlag.

[5] K. D. Electronic health record standards. volume 45, pages 136–144. Methods Inf Med, 2006.

[6] E. Denti, A. Omicini, and A. Ricci. Multi-paradigm Java-Prolog integration in tuProlog. *Sci. Comput. Program.*, 57(2):217–250, 2005.

[7] A. Dogac, G. Laleci, T. Aden, and M. Eichelberg. Enhancing IHE XDS for Federated Clinical Affinity Domain support. *IEEE Transactions on Information Technology in Biomedicine*, 11(2):213–221, 2007.

[8] A. Dogac, G. B. Laleci, S. Kirbas, Y. Kabak, S. S. Sinir, A. Yildiz, and Y. Gurcan. Artemis: deploying semantically enriched web services in the healthcare domain. *Inf. Syst.*, 31:321–339, June 2006.

[9] R. H. Dolin, L. Alschuler, S. Boyer, and C. Beebe. An update on hl7Êijs xml-based document representation standards. *Proceedings of the AMIA Symposium*, pages 190–194, 2000.

[10] D. Fensel. Triple-Space Computing: Semantic Web Services Based on Persistent Publication of Information. In F. Aagesen, C. Anutariya, and V. Wuwongse, editors, *Intelligence in Communication Systems*, volume 3283 of *Lecture Notes in Computer Science*, pages 43–53. Springer Berlin / Heidelberg, 2004.

[11] A. Geissbuhler, S. Spahni, A. Assimacopoulos, M. Raetzo, and G. Gobet. Design of a patient-centered, multi-institutional healthcare information network using peer-to-peer communication in a highly distributed architecture. *Medinfo*, 11(Pt 2):1048–52, 2004.

[12] D. Gelernter. Generative communication in Linda. *ACM Trans. Program. Lang. Syst.*, 7:80–112, January 1985.

[13] J. Grimson, W. Grimson, and W. Hasselbring. The SI challenge in health care. *Commun. ACM*, 43:48–55, June 2000.

[14] D. T. Gunter and P. N. Terry. The emergence of national electronic health record architectures in the united states and australia: Models, costs, and questions. *J Med Internet Res*, 7(1):e3, Mar 2005.

[15] J. Hendler. Agents and the Semantic Web. *IEEE Intelligent Systems*, 16:30–37, 2001.

[16] P. Hitzler, M. Krötzsch, and S. Rudolph. *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC, 2009.

[17] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From SHIQ and RDF to OWL: the making of a Web Ontology Language. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(1):7 – 26, 2003.

[18] IHE. IHE ITI white paper: Cross couminity information exchange, 2008. `http://www.ihe.net/Technical_Framework/upload/IHE_ITI_TF_White_Paper_Cross_Community_2008-11-07.pdf`.

[19] Integrating the Healthcare Enterprises. Technical framework integration profiles vol 1, 2011. `http://www.ihe.net`.

[20] T. Kifor, L. Varga, J. Vazquez-Salceda, S. Alvarez, S. Willmott, S. Miles, and L. Moreau. Provenance in agent-mediated healthcare systems. *Intelligent Systems, IEEE*, 21(6):38 –46, 2006.

[21] E. Nardini, M. Viroli, and E. Panzavolta. Coordination in open and dynamic environments with TuCSoN semantic tuple centres. In S. Y. Shin, S. Ossowski, M. Schumacher, M. J. Palakal, and C.-C. Hung, editors, *Proceedings of the 2010 ACM Symposium on Applied Computing (SAC)*, pages 2037–2044. ACM, 2010.

[22] L. J. B. Nixon, D. Cerizza, E. D. Valle, E. P. B. Simperl, and R. Krummenacher. Enabling Collaborative eHealth through Triplespace Computing. In *WETICE*, pages 80–85. IEEE Computer Society, 2007.

[23] A. Omicini. Formal ReSpecT in the A&A Perspective. *Electron. Notes Theor. Comput. Sci.*, 175:97–117, June 2007.

[24] A. Omicini and E. Denti. From tuple spaces to tuple centres. *Science of Computer Programming*, 41(3):277 – 294, 2001.

[25] A. Omicini and E. Denti. From Tuple Spaces to Tuple Centres. *Science of Computer Programming*, 41(3):277–294, nov 2001.

[26] J. Singh, J. Bacon, D. Weerasinghe, O. Akan, P. Bellavista, J. Cao, F. Dressler, D. Ferrari, M. Gerla, H. Kobayashi, S. Palazzo, S. Sahni, X. S. Shen, M. Stan, J. Xiaohua, A. Zomaya, and G. Coulson. *Event-Based Data Dissemination Control in Healthcare*, volume 0001, pages 167–174. Springer Berlin Heidelberg, 2009.

[27] F. Wozak, E. Ammenwerth, A. Hrbst, P. Sgner, R. Mair, and T. Schabetsberger. IHE based Interoperability - Benefits and Challenges. In S. K. Andersen, G. O. Klein, S. Schulz, and J. Aarts, editors, *MIE*, volume 136 of *Studies in Health Technology and Informatics*, pages 771–776. IOS Press, 2008.

Agent-Supported Assessment for Personalized Ambient Assisted Living

Helena Lindgren
Department of Computing Science
SE-901 87 Umeå
Sweden
helena@cs.umu.se

Farahnaz Yekeh
Department of Computing Science
SE-901 87 Umeå
Sweden
yekeh@cs.umu.se

Jayalakshmi Baskar
Department of Computing Science
SE-901 87 Umeå
Sweden
jaya@cs.umu.se

Chunli Yan
Department of Computing Science
SE-901 87 Umeå
Sweden
chunli@cs.umu.se

ABSTRACT

Existing approaches to ambient assisted living (AAL) often fail to consider a human agent's needs from a holistic perspective. In particular the regular assessment of their changing abilities, skills and limitations are often treated as a separate matter in healthcare, thereby affecting the possibilities to provide support tailored to their current condition. Therefore, the objective of this work is to integrate assessment done by the healthcare professional into the framework of AAL. We use a case scenario in the collaborative development with domain experts to demonstrate and develop the interaction between software agents and with the older adult in assessment and adaptation for supporting him/her in a home environment. The scenario also serves as an outline for a requirements analysis of the formal agent-based dialogues to be implemented. The results include a partial implementation of the scenario done by domain experts in their use of a semantic web-based knowledge and interaction modelling environment for domain professionals (ACKTUS). The resulting prototype applications are exemplified in a description of the scenario and an initial prototype implementation of selected agent-based diagnostic dialogues is presented.

Keywords

personalization; knowledge-based systems; agents; assessment; argumentation-based dialogues; ambient intelligence; smart environments; ubiquitous computing; pervasive healthcare

1. INTRODUCTION

In order to equip an individual with computer-based support in daily living for increasing autonomy, security, health, social inclusion and quality of life, a holistic view on the individual's situation needs to be adopted. The wishes, needs and abilities of the individual need to be assessed in order to optimize the design of the tailored support. Furthermore, the assessments need to be done continuously, in order to adjust the tailored support to changing needs, abilities, wishes and contextual factors. Therefore, we integrate assessment into a framework for developing and maintaining ambient assisted living, with a combination of methods where the professional assessment done by health care professionals, partly accomplished by agent-based dialogues, plays a key role [1]. The results of these assessments determine wether a person would benefit from using, or continuing using the system, and how it should be tailored to changing needs, e.g., in what way the individual best interacts with the system and what kind of support should be provided.

The framework proposed in [1] synthesized, and was built upon results from 1) a case study where eight older adults were using tailored web interfaces for activity support during two months, where the content and interaction design were based on occupational therapists assessments [2], 2) the work on developing a semantic web-based knowledge and interaction modelling environment for domain professionals (ACKTUS) [3, 4], and 3) the work on developing an adopting an ego-centric interaction model for assessing the interaction environment [5, 6, 7]. These prototype systems aim to collaborate by means of a service-oriented architecture through the proposed multi-agent system and constitute an ambient assisted living (AAL) home environment that serves the purpose to support and maintain a holistic user-centred view of an individual's life situation. The tailoring of the AAL environment to an individual is ideally based on a combination of automated methods for skills and ability detection, activity recognition and evaluation with the professional's regular assessment of the individual's ability, needs and wishes to perform different activities (e.g., ADL and leisure activities [8, 9]). The professional's assessment provides information about how interactive systems should be adapted to the individual, e.g., based on physical, cognitive, social and/or psychological limitations and abilities.

The purpose of the work presented in this paper is to

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), June, 4–8, 2012, Valencia, Spain.

develop a multi-agent system that takes knowledge repositories developed using ACKTUS into use in dialogues between agents and with human users. The contributions of this paper are the following: an architecture of the multi-agent system basing the functional requirements on a case scenario and a persona; an extension of an inquiry dialogue system for improving human agents' interaction with software agents; and a prototype implementation of selected agent-based diagnostic dialogues. For the purpose of this paper, we assume that the activity recognition system feeds information into the knowledge bases about the individual, to be used in assessments.

The paper is organized as follows. After our case scenario is described, we focus on the dialogues between human and software agents that utilize ACKTUS knowledge repositories. ACKTUS is described in subsequent section, followed by a section where ACKTUS is extended with agent-based argumentation dialogues. An initial prototype implementation of selected diagnostic dialogues is presented. A section reviewing related work is presented and the results are discussed.

2. THE RUT PERSONA AND SCENARIO

We focus our requirements analysis on a *persona*, an archetype of a potential user and a case scenario [10, 11]. The main obvious benefit of applying this approach is that it allows for sharing a common goal scenario among all professionals participating in the development and across professional domains. The persona functions as a knowledge artefact together with the knowledge artefacts in the form of support applications mediating the different professional domain's knowledge in the development process. Our scenario is a description of how the anticipated use situations may proceed when technology is developed to support the situations [12]. The main scenario is designed to capture a holistic perspective on assessment in a use situation (i.e. from a health care professional's perspective), which makes our scenario different from e.g., [13]. However, the intervention part gives also detailed descriptions of the desired activities to be performed from the perspective of the older adult. Moreover, we use the persona and scenario for specifying the functional requirements of the multi-agent system, which will be the focus in this paper, and for evaluating the developed support applications (a pilot study of the development driven by domain professionals and evaluation is presented in [14]).

Our scenario is based on an authentic case of an older woman who suffered from a few falls before a hip fracture became the result with a long hospitalization with severe anxiety, delirium and fatal complications as a consequence. We envision a different scenario and outcome by introducing the AAL environment as a supplementary intervention to better meet her changing needs. This case shares needs and wishes also with some of the participants in an earlier case study and is therefore considered representative [2]. The time period in focus spans over 8 months, initiated by a visit by the occupational therapist (OT) for the initial assessment. The reason for initiating the contact with the OT was not cognitive impairment, but the question was risen by the son. Interventions were decided upon and realized. Support applications are envisioned to be instrumental in the interventions, partly by enabling a continuous follow-up. We explore and describe the case study as five distinct but related activities: 1) initial assessment, 2) referral to physician for assessing a potential dementia, 3) determine interventions, 4) apply interventions in the daily life of the individual including a continuing assessment (and follow-up), and 5) a renewed assessment (Figure 1). In this paper we set focus on the dialogues envisioned between human and software agents. In the following subsections the different phases of the scenario is described, in addition to the different actors and how the knowledge-based support applications created by the domain experts are integrated in the scenario (*I-rehab* for OTs, *I-dementia* for physicians, *I-care* for nurses and *I-Help* designed for the older adult and relatives, see Figure 1). The user interface implemented for evaluating the dialogues to be conducted with the older adult is text-based with structured answer alternatives. However, in the future the content of dialogues can be medicated by other modality than text, tailored to the individual. The primary software agents identified are the *Domain Agents* (DA) representing different knowledge domains and the *Coach Agent* (CA). They are further described in Section 4.1.

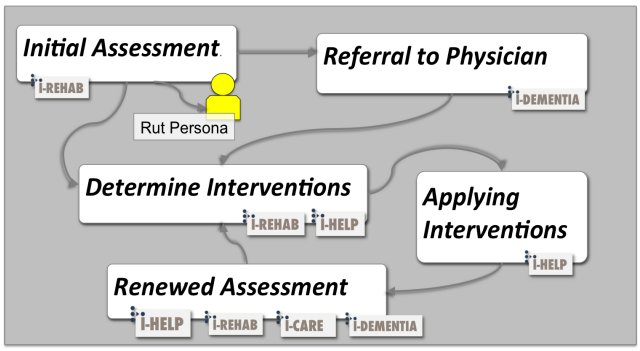

Figure 1: The Rut scenario and its prototype support applications.

2.1 Initial Assessment

The first dialogue is essentially an *information seeking* dialogue and involves OT, Rut and her son in their first encounter. The roles of the actors are basically that the OT asks and the others assert information, typically by selecting among a set of alternatives. Two supplementary instruments for assessing instrumental activities of daily living (iADL) and a simplified version of an interest checklist were used. All were accessible through the I-Rehab application, which is designed as a knowledge-based support system for the OTs. During the interview with Rut the OT used the web-versions of the instruments to note what Rut and her son were describing. This usage resulted in the knowledge repository about Rut that is later used in followup dialogues. This dialogue was also performed mediated by I-help, with the intention that Rut and her son could interact with the support system.

2.2 Determine Interventions

The part of the assessment where goals for intervention are identified is a critical part. This part can be seen as a dialogue with the goal to decide upon which actions to be performed. The following two types of questions form the base for defining goals: 1) How important is it for you to do activity A?, and 2) How satisfied are you with how you are able to do activity A? When an activity is identified as very

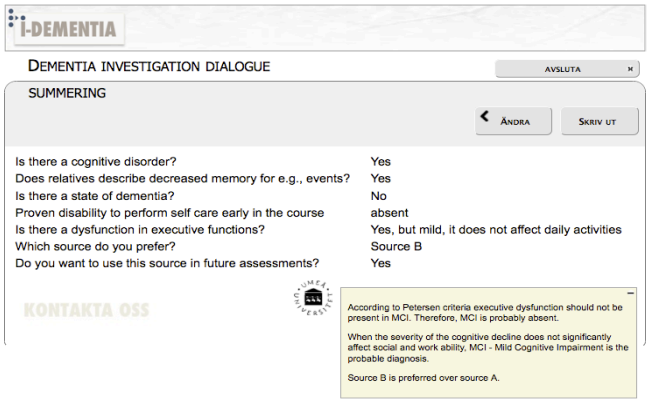

Figure 2: Summary of diagnostic dialogue as modelled by domain experts.

important and troublesome to perform, the reasons for the difficulties are identified in an information seeking dialogue. When an activity is judged being suitable to support by computerized means, I-Help comes into play as an intervention proposed by the OT. The tailoring of her I-Help application is based on a particular ACKTUS assessment protocol defined for the purpose, which underlies the dialogue that the system/OT has with Rut about the intervention.

2.3 Referral to Physician

Since Rut's son described difficulties with cognitively demanding tasks an assessment was done of whether Rut had an emerging dementia disease. In this assessment the physician used the application I-Dementia as a guide for the assessment. The diagnostic dialogue that takes place is of the type *inquiry* dialogue, the goal is to create new knowledge. Medical domain experts modeled the content and interactivity of this dialogue and part of the result is shown in Figure 2 [14]. In this work we use it as an example of the implementation of our agent system in Section 4.3 and details can be found in Table 1.

The data about Rut was analysed and a disagreement arose about which knowledge source to apply. The Physician suggested that Rut has mild cognitive impairment (MCI) (too mild to fulfill the criteria for dementia) based on a different source than the one the Domain Agent (DA) uses to exclude also the presence of MCI. Therefore, the Coach Agent (CA) explicitly asks for the physician's preference in the matter to be applied in future cases.

2.4 Applying Interventions

In the intervention part of the scenario typical and targeted activities performed by Rut in her daily life with the support from tailored applications are described. For space reasons we omit this, and focus on the evaluation of the activities done by the CA through the I-Help application. For some of the questions defined by the OT there are follow-up questions to assess degrees and some types of answers generate advice. The questions concern the activities identified as troublesome in the initial assessment and the key follow-up questions are the same: 1) How important is it for you to do activity A?, and 2) How satisfied are you with how you are able to do activity A?. Since Rut also was worried about different things, questions about worries were also included and a question about her current view on her health.

2.5 Renewed Assessment

The CA detects a change in activity pattern based on information from the activity recognition system after a period, Rut starts to walk around nighttime. The information is contradicting earlier information, leading to further dialogues to resolve the conflict. This leads to an initiation of dialogues with other system agents (DAs for the different domains) to find reasons for conflicting information. In our scenario, after the dialogues between the software agents there is still insufficient information for resolving the reasons for Rut to be wandering about nighttime. This leads the CA to initiate a dialogue with Rut the next morning with the same topic (Disturbed sleep patterns), leading to questions about sleeping pills, worries, pain, stomach issues and incontinence, and nested dialogues with the purpose to ask Rut if she wants to have contact with a professional to discuss the potential reasons for disturbed sleep that come up during the dialogue. At the end of the dialogue, the CA also shows Rut a summary of the dialogue, and asks for Rut's permission to send the summary to the nurse as a preparation of a visit. The nurse who evaluated the dialogues pointed out that an essential part of the knowledge is common between the nursing and the rehabilitation domains, and they typically cooperate. Therefore, she would make use of parts of the rehab content of the prototype in addition to the care content if she would have met the client in her home for a followup [14].

3. ACKTUS

The knowledge-based prototype applications are developed using ACKTUS (activity-centred knowledge and interaction modelling tailored to users) (Figure 3) [4]. ACKTUS is an evolving semantic web application that is designed to allow domain experts who are typically not familiar with knowledge engineering to author and model the knowledge content of, and design the interaction with, knowledge-based applications. The purpose is to bridge the knowledge gap between medical experts and knowledge engineers, so that the domain experts can use ACKTUS for adjusting the tailoring of applications as part of the development of their daily practice. ACKTUS has emerged as a result from experiences in developing socio-technical systems for the medical and health domains [2, 15, 16, 17].

ACKTUS consists of a service-oriented architecture, which includes an RDF/OWL ontology, Sesame repositories and dedicated user interfaces (Figure 3). The different services to be provided by the system are currently being developed, among which the reasoning services can be accessed and utilized by the agent system outlined in this paper.

3.1 ACKTUS Ontology

The core ontology implements a generic model of activity [18, 3] and captures 1) components of reasoning in the form of an argumentation framework, 2) components used for tailoring interaction with the resulting knowledge applications and 3) components for modelling the user agents as actors in a situation where the application is used, e.g., for reasoning, performing daily activities, for entertainment or social interaction. The ACKTUS applications share this common core ontology, which is extended with specifics for each knowledge domain. Thus, for each knowledge domain

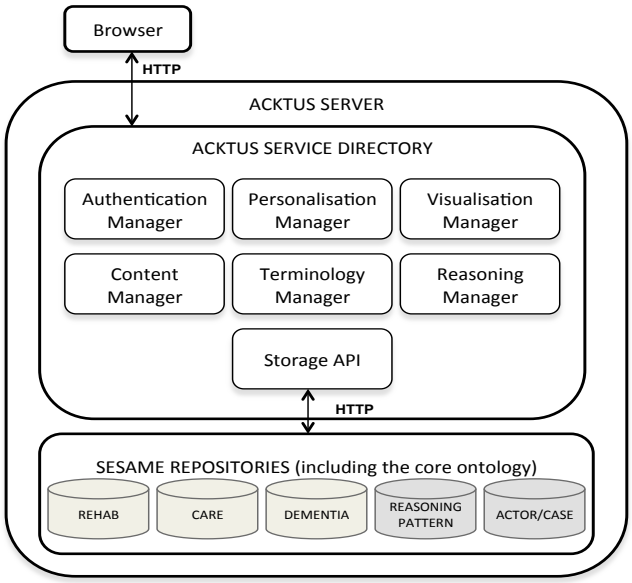

Figure 3: ACKTUS architecture

(care, rehabilitation, etc.) a domain knowledge repository is created in the modelling of knowledge, containing validated assessment instruments (e.g., [8, 9]), clinical practice guidelines (e.g., [19]), etc. Preference orders among knowledge sources, level of expertise in knowledge domains and other personal and contextual factors may be taken into account in adjustments made in a person's home environment for increasing ability in activity performance.

A key feature of the ontology is the concept system, to which each object that carries knowledge is associated. In the domain experts' modelling they also model the concept structure and identify key concepts for their domains. The basic structure of the concept system is common between the domain applications and is based on ICF (International classification of ability, functioning and health)[1] extended with diseases and syndromes. However, the representatives from the different knowledge domains refine branches of the concept tree differently, mirroring their different specialties. As in clinical practice where the different types of professionals supplement each other in daily knowledge work, these domain ontologies are supplementary. Moreover, the concept-system-node class serves as the data structure for the information about the user and/or patient/client. The concept system provides the terminology and semantics of expressions, allowing for an agent to specify a topic of a dialogue comprehensible to other agents.

Data to be fed into rules is collected by templates for the purpose, designed by the domain experts. These frame and define the content of the locutions in dialogues with the user. The templates are ordered by the user into possibly nested *assessment protocols*, used for representing structured data capture activity. While the reasoning contexts are suitable for supporting reasoning at a higher level of cognitive complexity, the assessment protocols are suitable for supporting lower level reasoning (e.g., data capture), typically as part of applying reasoning contexts.

The rules that are defined by the domain experts are simple structures with a set of claims as body and as head of the rule one conclusion (also a claim about a topic), an advice or an action (assessment protocol) to be activated. The underlying structure for rules, schemes and critical questions is based on the Argument Interchange Format (AIF) developed for facilitating sharing and visualization of arguments using the WWW [20].

[1]http://www.who.int/classifications/icf/en/

4. EXTENDING ACKTUS WITH TAILORED AGENT-BASED DIALOGUES

A major purpose of the domain-specific support applications developed using ACKTUS is to educate the user and provide new evidence-based knowledge at the point where it is useful. The goal is to improve the daily care of individuals (e.g., early detecting an emerging dementia disease). In this work we take the mediation of this formalized knowledge a step further, by developing a multi-agent system where the purpose is to challenge and support the professional. Moreover, in the same way as the professional is supported in their daily work, the older adult in their home can be supported in their daily activities. The motivation for using a multi-agent dialogue perspective in our work is that this approach allows for modeling the conditions for knowledge development in individuals as well as the system, in spite of ambiguous and incomplete domain knowledge. In addition, we anticipate that agent-based dialogues will facilitate more intuitive and natural dialogues between the user and the system.

The purpose of the agents' dialogues is to share (defeasible) knowledge to construct arguments supporting beliefs, and in this way increase each agent's knowledge repository (belief base). We adopt the BDI framework for agents, meaning that each agent has beliefs, desires and intentions. We assume that the participants are cooperative and reliable, aiming at disseminating and increasing knowledge and finding optimal decisions and actions in client cases. We assume also that one single medical actor does not possess all knowledge required for providing a client optimal care, but that the knowledge is distributed over a team of professionals with different viewpoints of a clinical situation. Indirectly, the implemented knowledge in the clinical decision-support system (CDS) represents the knowledge of an expert in the domain, without assuring that this expert possesses all knowledge relevant for a clinical situation (which is also the case in real clinical situations).

The dialogues can be activated by any of the agents, and the content of messages is selected from their belief bases, based on the topic for the dialogue [21]. The topic is framed by the underlying ontology, which defines the reasoning context and associated critical questions, rules, knowledge sources, etc. [3]. Critical questions function as locutions in dialogues. A dialogue can also involve the human agent, then visible through the I-help application and possibly through audio modality. A topic needs to be selected, possibly among predefined topics visible through a graphical user interface, or by vocal interaction. In this paper we assume that information about the activity patterns is collected by the egocentric interaction model for assessing the interaction environment [5, 6] and that this information is interpreted and fed into the client repository and belief base of a Coach Agent through a service directory.

In the following subsections types of dialogues, their formal properties and functional requirements will be described (e.g., agent roles, rules for speaker order, termination rules,

commitment stores and belief bases). Furthermore, an initial prototype implementation of the diagnostic dialogues involving a physician, the Domain Agent and the Coach Agent is described and evaluated.

4.1 Agent Roles

In our framework a Mediator Agent (MA) organizes the dialogues between different application agents, where knowledge can be used between knowledge repositories, depending on the topic to investigate and on users' authorization. Apart from a Mediator Agent, which functions as a service provider and does not contribute to nor affect the content of dialogues, a Coach Agent and a Domain Agent are activated in the interaction with an ACKTUS application (Figure 4).

The Coach Agent (CA) acts on behalf of the user, represents the user and guards the user's interests. It detects what the human user agent does in interaction with the application, organizes and updates the user's belief base, and acts when the user does not follow previously stated preference orders or pattern of interaction (partly represented in the user's belief base). The CA also organizes the user model, which collects the information about the user's limitations and resources (e.g., physical, cognitive, social) and preferences about e.g., who has access to the information. In our scenario the CA detects pattern of activity as well as deviations from pattern of activity in an individual user partly based on information from the personal activity-centric middleware [7].

The Domain Agent (DA) acts as a domain professional, making use of the domain knowledge repository. In the interaction with the CA and the therapist user in the assessment application, the DA provides the expert perspective to reasoning, when possible based on evidence-based guidelines and validated assessment instruments for the purpose to educate the therapist and support daily work with e.g., older persons who need adjustments in their home environment. DA represents a domain professional in the dialogues with CA, and CA represents the individual. DA may also act as the domain professional in interaction with the individual through the I-Help application.

4.2 Dialogue Types, their Formal Properties and Design

Dialogue games are commonly used to describe and characterize argument-based dialogues involving one or more agents (e.g., [22]). Dialogue games are typically organized by a limited set of allowed acts, or moves, with rules (representing a protocol) directing how the moves can be done at each point in the dialogue, the outcome of a move, and when a game is terminated. The purpose of a dialogue game can be different, corresponding to the motives the agent or agents have with their participation.

We analysed the different dialogues exemplified in our scenario, interpreted them as dialogue games, and categorised them into three of the types of dialogues described by Walton and Krabbe [23]: *information seeking, inquiry* and *deliberation dialogue*. They differ by their purpose where an *information seeking* dialogue aims at collecting information, *inquiry dialogues* aim at collaboratively create new knowledge and a *deliberation dialogue* aims at collaboratively decide upon a plan of action to be performed.

For each of the dialogues we also identified the different purposes of moves (locutions) that the agents make.

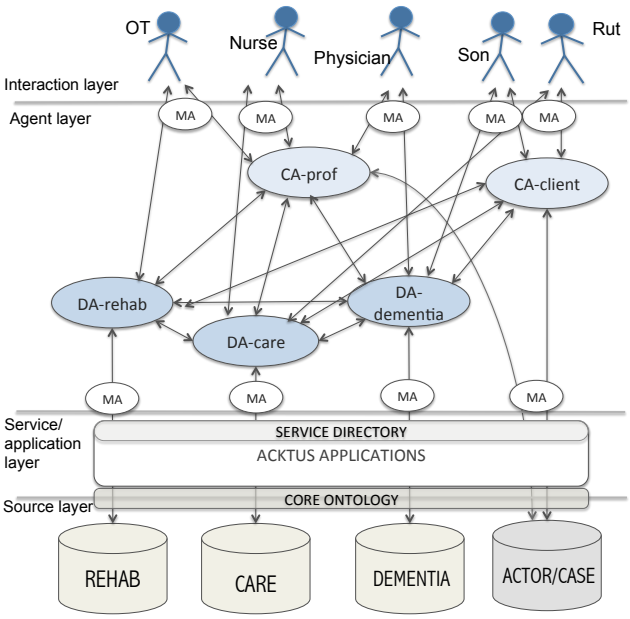

Figure 4: Extension of the ACKTUS architecture with an agent layer based on the case scenario.

The following purposes were considered a minimal set necessary to execute the dialogues in our scenario: *open/initiate, close/terminate, ask* and *assert*. For illustrating the dialogues in our use scenario in the development sessions with domain experts, we use an algorithm for executing the information seeking dialogues and for simulating the inquiry and deliberation dialogues. The algorithm only makes use of ACKTUS assessment protocols, their content, their associated rules and their consequents as dialogue flows structured by the domain experts in the modelling sessions. In this highly structured form, the autonomy of agents is limited to a minimum. However, it serves as a starting point for our user-driven approach to development of agent-based dialogues. The domain professionals are in control of the agent's behavior. The algorithm is as follows:

1. Identify the assessment protocol and add its ordered contents to the list of items that need to be executed
2. For each item in the list:
   (a) Deliver the message and add the user's response as a new belief to the belief base
   (b) Match the new belief with the ACKTUS rules and ACKTUS information, then compare the result of matching to:
      i. If the result is a new assessment protocol, start the execution of this assessment protocol (activity)
      ii. If the result is a new advice, show the advice immediately and/or save it for showing in summary page
      iii. If the result is a conclusion, add the conclusion (the new knowledge) to the belief base and to the case repository

3. Show the summary page including conclusions, advices, asked questions and the user's responses (see example in Figure 2)

The same building blocks created using ACKTUS and executed in the simulated agent-dialogues, are used in the agent-based dialogues. Consequently, we can accomplish information-seeking dialogues by using the ACKTUS assessment protocols, since they are designed for the purpose. We can also reuse the protocols for reasoning about which actions to make, when the aim is to investigate and collect more information to be used in reasoning about topics. The topics are identified using the ACKTUS concept system or by critical questions, which define reasoning contexts. For the implementation of the agent-based dialogues of the inquiry type, we adapt the approach developed by Black and Hunter [22], further described in the following subsection.

4.2.1 Inquiry Dialogues

In [21] an inquiry dialogue system developed by Black and Hunter [22] was applied for capturing diagnostic dialogues concerning dementia. The formal protocols for dialogues are adopted from [22] and are extended with values and preferences drawn from the ACKTUS argumentation framework and knowledge repositories. The work presented in [22] is adapted to ACKTUS mainly in that we consider individuals' preferences as central for personalization and tailored feed-back in a learning process. We also make use of the ACKTUS *reasoning contexts* to identify the topic of a dialogue (a concept identifiable in the ACKTUS ontology) and to activate a corresponding subset of the knowledge base to form the belief base of the CDS agent. This way the dialogues can follow the user's trail of thinking in that the user choses topics, e.g., in a differential diagnostic reasoning process. Our adaptation will be described from the perspective of a practical diagnostic example, and the interested reader finds the formal definitions in [22].

In the following summative description of inquiry dialogues, we apply Black and Hunter's distinction between two sub-types of warrant dialogues; *warrant inquiry dialogue* and *argument inquiry dialogue*. The difference is that the outcome of a warrant inquiry dialogue is a dialectical tree with argument nodes marked as defeated or undefeated, while the argument inquiry dialogue builds the argument to be fed into the warrant inquiry dialogue. Their approach limits the set of possible moves to *open, close* and *assert*. Therefore, for our purposes we add the move *ask* whenever there is a reason for an agent to ask instead of making a close move. We also apply the rules for dialogue execution specified in [22]. To terminate a dialogue, all agents should make a close move. It is allowed to have argument inquiry dialogue nested within warrant inquiry dialogues, but not vice versa. Argument inquiry dialogues can be performed within other argument inquiry dialogues. In none of the cases an agent is able on its own to construct the argument or the dialectical tree, based on its limited set of beliefs. All agents take turn in making moves in the dialogue.

A major difference between the two types of inquiry dialogues is that in an argument inquiry dialogue the agents are not allowed to determine the acceptability of the arguments constructed, while in the warrant inquiry dialogue determining acceptability is the purpose. Therefore, argument inquiry dialogues are often embedded within warrant inquiry dialogues. Another difference is that the topic of an argument inquiry dialogue is a defeasible rule, while the topic of a warrant inquiry dialogue is a defeasible fact. There are three legal moves defined in [22] for the two types of dialogue: *open* ($\langle x, open, dialogue(\theta, \gamma)\rangle$), *assert* ($\langle x, assert, \langle \Phi, \phi\rangle\rangle$) and *close* ($\langle x, close, dialogue(\theta, \gamma)\rangle$). The format used for moves in the example dialogue in Table 1 follows the format described in [22], where x represents the agent, $\langle \Phi, \phi\rangle$ is an argument, $\theta = wi$ and γ represents a defeasible fact for a warrant inquiry, and in the case of an argument inquiry $\theta = ai$ and γ represents a defeasible rule. In our example we add the move *ask* in the following format: $ask\,(\langle x, ask, \langle \gamma, cq_y\rangle\rangle)$, where γ represents an unknown defeasible fact. Each agent has a possibly inconsistent belief base and it is assumed that all agents have the same role [22]. By making a query store (which is loaded with for the topic relevant sub-topics) and each agent's commitment store (loaded with asserted knowledge during the dialogue) public, the agents can make use of common knowledge in the dialogue. A dialogue is terminated when all participants have made a close move, which guarantees that all relevant information has been taken into consideration.

Black and Hunter provide a protocol for modeling inquiry dialogues and a strategy for generating dialogues (choosing among candidate moves in a dialogue), which uses an adapted version of Garcia and Simari's Defeasible Logic Programming (DeLP) for representing agents' beliefs [24]. DeLP is adapted by making the sets of strict rules and facts empty and define a defeasible fact. This way all knowledge becomes defeasible, which is suitable for their as well as our purposes. A defeasible rule is denoted $\alpha_1 \wedge ... \wedge \alpha_n \rightarrow \alpha_0$ where α_i is a literal for $0 \leq$ i $\leq$ n. A defeasible fact is denoted α where α is a literal.

Black and Hunter also associate a preference level with a defeasible rule or fact in the formation of a belief, although they do not account for the source of this preference level. The preference ordering is used in the comparison of two arguments. We replace the numbers used in [22] with an explicit ranking among knowledge sources to make the comparison in our example transparent and associate this ranking to defeasible rules. Furthermore, we use an ordered set of values to associate strength to defeasible facts. A belief is a pair (ϕ, S) where ϕ is either a defeasible rule or a defeasible fact. If ϕ is a defeasible rule then S $\in S_0 = \{cpg, cons, rot\}$ and if ϕ is a defeasible fact then in our example S $\in S_1 = \{present, unknown, absent\}$. Clinical practice guidelines (*cpg*) are considered more reliable than consensus guidelines (*cons*), while both are considered more reliable than a 'rule-of-thumb' (*rot*), which is often based on fragmented experiences of an individual professional [21]. Therefore, the following additional beliefs about the strength of knowledge sources are integrated in the CDS agent's belief base: ($cpg > cons$, $cpg > rot$, $cons > rot$), where $>$ is a binary relation meaning 'strictly preferred to'. In our example given in Table 1 two comparable sources of the same type is used (Sources A and B), and the physician's preference directs the final outcome. Moreover, in our example dialogue the Physician (A_1) and the DA (A_2) reuse the following beliefs stored after earlier dialogues about Rut: {(Dementia, *absent*), (CogDis, *present*)}. When the dialogue has ended, a dialectical tree can be formed, where arguments can be evaluated as being *Defeated* or *Undefeated*. In our example two contradicting outcomes can be found, using rules based on two different sources. Since the sources are equally

Table 1: Example of a warrant inquiry dialogue with nested argument inquiry dialogues, where *t* represents timepoints, CS is commitment store and m_t is a move at timepoint *t*.

t	A_1 CS	m_t	A_2 CS
1		⟨1, *open*, wi_1 (MCI, *present*)⟩	
2		⟨2, *open*, ai_1 ((CogDis, *present*) ∧ (Dementia, *absent*) ∧ (ADL, *absent*) ∧ (Exec, *absent*) → ⟨MCI, *present*⟩), Source C⟩	
3	(CogDis, *present*)	⟨1, *assert* ⟨{(CogDis, *present*)}, (CogDis, *present*)⟩⟩	
4		⟨2, *ask* ⟨{(ADL, *unknown*)}, Trouble with self care?⟩⟩	(CogDis, *present*)
5	(ADL, *absent*)	⟨1, *assert* ⟨{(ADL, *absent*)}, (ADL, *absent*)⟩⟩	
6		⟨2, *ask* ⟨{(Exec, *unknown*)}, Executive dysfunctions?⟩⟩	(ADL, *absent*)
7	(Exec, *mild*)	⟨1, *assert* ⟨{(Exec, *Mild*)}, (Exec, *Mild*)⟩⟩	
8		⟨2, *close*, ai_1 ((CogDis, *present*) ∧ (Dementia, *absent*) ∧ (ADL, *absent*) ∧ (Exec, *absent*) → ⟨MCI, *present*⟩)⟩	(Exec, *mild*)
9		⟨1, *close*, ai_1 ((CogDis, *present*) ∧ (Dementia, *absent*) ∧ (ADL, *absent*) ∧ (Exec, *absent*) → ⟨MCI, *present*⟩)⟩	
10		⟨2, *open*, ai_2 ((Exec, *mild*) → ⟨MCI, *absent*⟩), Source C⟩	
11	(MCI, *absent*)	⟨1, *assert Arg1* ⟨{(Exec, *mild*), ((Exec, *mild*) → ⟨MCI, *absent*⟩)⟩}, (MCI, *absent*), Source A⟩⟩	
12		⟨2, *close*, ai_2 ((Exec, *mild*) → ⟨MCI, *absent*⟩)⟩	(MCI, *absent*)
13		⟨1, *close*, ai_2 ((Exec, *mild*) → ⟨MCI, *absent*⟩)⟩	
14		⟨2, *close*, wi_1 (MCI, *present*)⟩	
15		⟨1, *open*, ai_3 ((CogDis, *present*) ∧ (Dementia, *absent*) ∧ (ADL, *absent*) ∧ (Exec, *mild*) → ⟨MCI, *present*⟩), Source D⟩	
16		⟨2, *assert Arg2* ⟨{(CogDis, *present*), (Dementia, *absent*), (ADL, *absent*), (Exec, *mild*), (CogDis, *present*) ∧ (Dementia, *absent*) ∧ (ADL, *absent*) ∧ (Exec, *mild*) → ⟨MCI, *present*⟩), Source B⟩⟩	(MCI, *present*)
17	(MCI, *present*)	⟨1, *close*, ai_3 ((CogDis, *present*) ∧ (Dementia, *absent*) ∧ (ADL, *absent*) ∧ (Exec, *mild*) → ⟨MCI, *present*⟩)⟩	
18		⟨2, *close*, ai_3 ((CogDis, *present*), (Dementia, *absent*), (ADL, *absent*), (Exec, *mild*) → ⟨MCI, *present*⟩)⟩	
19		⟨1, *close*, wi_1 (MCI, *present*)⟩	
20		⟨2, *assert Arg3* ⟨{(Source A > Source B), Arg1, Arg 2}, (MCI, *absent*)⟩⟩	(A>B)
21	(A>B) (B>A)	⟨1, *assert Arg4* ⟨{(Source B > Source A), Arg1, Arg 2}, (MCI, *present*)⟩⟩	
11		⟨2, *close*, wi_1 (MCI, *present*)⟩	(B>A)
12		⟨1, *close* wi_1, (MCI, *present*)⟩	
	conclusion:	Resulting dialectical tree: (Claim - Argument 0: MCI present): Undefeated -: (Argument 1: MCI absent): Defeated -: (Argument 2: MCI present): Undefeated -: (Argument 3: MCI absent): Defeated -: (Argument 4: MCI present): Undefeated.	

Abbreviations

CogDis: Cognitive disorder
Exec: Executive dysfunction
ADL: Activities of daily living
MCI: Mild Cognitive Impairment

reliable considering their types, we include the preference orders of both the DA and the Physician. Since the physician is responsible for the diagnostic decisions to be made, the Physician agent in our example overrules the DA's preferences (Table 1).

4.3 Implementation of Dialogues about Medical Diagnosis

An initial prototype implementation of the inquiry dialogues described in 2.3 was developed using the JADE (Java Agent Development Framework) platform [25].

Table 2: Algorithm for dialogue implementation

1	**Initiating Agent:**	
	Selects and sends the topic of dialogue	$\langle open \rangle$ wi
2	**Agent:**	
	Receive and Check MessageType	
	If RecMessageType==$\langle assert \rangle$	
	Update CS	
	If RecMessageType==$\langle ask \rangle$	
	SelectNextMove → *assert/close*	
	Else	
	SelectNextMove → *open/assert /ask/close*	
	If NextMove==$\langle open \rangle$	
	ai=SelectDefeasibleRule	
	Put the literals of the rule in the QS and send the topic	$\langle open \rangle$ ai
	Wait to receive next message	
	If NextMove==$\langle assert \rangle$	
	Construct the argument: collect the set of grounds that supports the claim (beliefs)	
	arg = SelectArgument	
	Send the argument	$\langle assert \rangle$ arg
	Wait to receive next message	
	If NextMove==$\langle ask \rangle$	
	Based on unknown facts in QS	
	cq= SelectCriticalQuestion	
	Send the critical question	$\langle ask \rangle$ cq
	Wait to receive next message	
	If NextMove==$\langle close \rangle$	
	Send close request	$\langle close \rangle$ ai/wi
	If RecMessageType==$\langle close \rangle$	
	Go to Step 3	
	Else	
	Wait to receive next message	
3	**Conclude**	

The Physician and the DA have their own belief bases. The Physician's belief base consists of observations and other knowledge about Rut, and rules (possibly rules-of-thumb or clinical guideline-based), while the DA's belief base is obtained from ACKTUS. The belief base consists of defeasible rules associated to schemes and critical questions stored in ACKTUS repositories.

A dialogue may be initiated by the Physician or the DA and the topic is sent to the other one. Agents participating in the dialogue apply a priority order in the selection of moves, namely *open*, *assert*, *ask* and finally *close*. By prioritizing *open* and *assert*, new knowledge will be created, which serves the purpose of supporting learning in the end user. However, if the received message is of the type *ask*, the responding agent is required to select between the two moves *assert* and *close*. The purpose of the *ask* move is to switch the burden of proof to another agent. Each agent has a Commitment Store (CS), which is updated with new beliefs when any (including itself) agent asserts an argument. A dialogue might include a Query Store (QS) which stores the literals of the rule when an argument inquiry dialogue is opened.

The algorithm for conducting the argumentative dialogue between these agents is presented in Table 2. When the application has been started, the participant agents are instantiated. One of the agents (Physician or DA) starts the dialogue by sending the topic for conversation to the other one (Step 1). Then, the receiver agent selects and performs its move and sends it to the initiator. Similarly, it is the initiator's turn to perform the execution and specify its move. They continue until both of them send the *close* moves respectively. Finally, it is time to conclude the dialogue (Step 3). At this stage the dialectical tree is constructed by the DA, which serves as a conclusion to be presented to the Physician agent.

The selection process for each move in each agent is as follows. The agent checks its belief base and commitment store. Then, it replies with a move with the type and the content which are selected by its *SelectNextMove* and *SelectDefeasibleRule*, *SelectArgument* or *SelectCriticalQuestion* functions. After sending the message, the agent waits to receive a message from the other one. If each agent has no more rules for opening argument inquiry dialogues related to the topic, no new knowledge to assert and no more question to ask, it closes the dialogue.

The algorithm is generic in order to be applicable in a situation when more than two agents participate in the dialogue. Participating agents must have a predefined order to send their messages, according to their turn and the dialogue is concluded and terminated after all of them have made a close move.

5. RELATED WORK

Formal argumentation frameworks and argumentation schemes have the potential to improve the reasoning and communication capabilities within and between software agents (e.g., [26]). In a system that contains different agents, each might have different knowledge. One agent can also have conflicting information contained in a belief base. These agents with their different expertise may collaborate and utilize their individual knowledge, participate in the reasoning process to make a decision or achieve a common goal. In some frameworks there are also mechanisms that allow an agent to learn from dialogues.

There are several proposals of formal frameworks for capturing the factors influencing an argumentation-based dialogue, and for resolving which arguments are valid and which are defeated. The most influential is the Dung semantics of argumentation frameworks, which have been extended and applied in many example cases, implementations of reasoners and argumentation software for different purposes (e.g., [24, 27]). Another example is the formal framework presented by Black and Atkinson [28] combining inquiry dia-

logue argumentation and practical reasoning (i.e., reasoning about actions). The combination of these approaches makes the reasoning easier among the agents with different perspectives on a subject. They have taken a formal specification of the two mentioned types of dialogues, extended it with additional critical questions, which make it possible to make more arguments while trying to find some consensus. The critical questions are generic, which is a difference from our work where the knowledge and associated questions are mainly domain-specific. However, it will be investigated in future work in what way our agent protocols conform to their generic framework. The implementation of their framework was also done using JADE (Java Agent Development Framework) [25].

Other approaches take a real world situation as starting point and frame the need for agent-based support. SHARE-IT [29] (Supported human autonomy for recovery and enhancement of cognitive and motor abilities using information technologies) implements a combination of multi-agent system and other techniques to aid the elders based on user scenarios [13]. The agents hold information about all physical devices that exist in the environment to control daily living activities and the conceptual world's information. The multi-agent system (MAS) includes a *patient agent, vehicle agent, caregiver agents, environment agent* and a *home agent* that monitor users and their activities, and manage their profiles. A difference in our work is that we integrate assessment into the context of use, and aim at using the professionals' instruments for continuing followup supported by agents.

Tolchinsky and coworkers presented an argumentation framework with heterogeneous agents to argue about the viability of transplantation of a human organ [30]. The purpose of their work is to improve the process of transplantation. The argumentation type supported in their work is a deliberation dialogue, since the participating agents collaboratively decide upon viability of human organs for transplantation. A mediator agent directs the deliberation and based on the arguments presents the final decision. The software agent uses argumentation schemes and critical questions submitted by doctors in order to direct exchange of argumentation among human and/or software agents [30]. The argumentation is done between a donor agent (an agent representing the hospital where the donor is located) and a potential recipient. In order to mediate the protocol based exchange of arguments between donor and recipient agents (that are assumed to be human doctors), a software agent named mediator agent is involved. As the arguments submitted by the agents and the dialogue ended, the mediator agent created the conversation graph and evaluates the argumentation. The mediator agent references the knowledge sources (a knowledge base of acceptability criteria, case base of earlier patient cases, and the reputation of the agent) to determine the validity of the submitted arguments and the strength of the argument. The approach by Tolchinsky et al. shares features with the work presented in this paper. They also used a dedicated web-based user interface for the domain experts to model the medical knowledge and they utilize agents with different roles to contribute to the decisions about actions. The main difference is that they focus on one particular decision problem where they utilize deliberation dialogues, while our scenario spans different decision situations that require different types of dialogues. The conversation-based protocol that mediator agent uses to direct submissions of arguments by donor and recipient agents, has been implemented using COGENT [31].

6. DISCUSSION

A persona and a case scenario have been used for outlining the requirements of a multi-agent system for argumentation-based dialogues. The main location for the activities in the scenario is the home of the older adult. The home environment integrates in our scenario ambient assisted living in the form of an activity recognition system that interacts with the agents of a knowledge-based system. Characteristics of the dialogues taking place at different points in the envisioned scenario were identified, such as types of dialogues, types of moves, which agent takes initiatives, content and structure of locutions.

In [21] an inquiry dialogue system developed by Black and Hunter was applied for capturing diagnostic dialogues concerning dementia. However, their approach limits the set of moves to open, close and assert. In interaction with human agents the possibility to pose questions is essential to achieve a dialogue that is perceived as natural by the human. Therefore, we add the move *ask* to our example scenario as the practical outcome when a knowledge-seeking (curious) agent does not know and can only make a close move otherwise. A dialogue about diagnosis defined in the scenario was implemented using JADE featuring the designed MAS utilizing ACKTUS knowledge repositories.

The development of the support application for the older adult targets other central issues, both with respect to ethical and privacy issues, but also the challenge to optimise personalisation and adaptability to changing abilities and needs in the individual. We have addressed these issues in our work, and outlined a possible scenario where the support applications may adapt to changes. Ongoing and future work includes finalizing the user-driven development of the knowledge-based support systems, the agent-based dialogue implementation, its formal framework and a suitable interaction design for the older adults as end users. Moreover, the activity recognition system will be integrated in a demonstrator environment so that the scenario can be evaluated in practice with older adults. Combining the activity recognition system with the knowledge-based support applications, the resulting support environment has the potential to support activity of importance to the individual, as compared to the majority of the existing approaches to AAL. Typically AAL environments are designed to control the physical surroundings and detect hazardous events such as falls, and activity at a basic operational level (types of movement, location, eating, sleep, etc). By adding meaning to activities at a higher level defined by the older adult, the older adult also becomes the designer and may get a sense of control over the AAL environment. This is an essential part of a system aiming at increasing an individual's autonomy and sense of coherence in daily life.

7. REFERENCES

[1] Lindgren, H., Surie, D., Nilsson, I.: Agent-Supported Assessment for Adaptive and Personalized Ambient Assisted Living. In: JM. Corchado, J Bajo, K Hallenborg, P Golinska and R Corchuelo (Eds.) Trends in Practical Applications of Agents and Multiagent Systems, pp. 23-32. Springer 2011.

[2] Lindgren H, Nilsson I (2009) Designing Systems for Health Promotion and Autonomy in Older Adults. In proc. Interact'09, LNCS 5727:700-703 Springer Berlin/Heidelberg

[3] Lindgren H, Winnberg P (2010) A Model for Interaction Design of Personalised Knowledge Systems in the Health Domain. In M. Szomszor and P. Kostkova (Eds.): E-Health 2010, LNICST 69, pp. 235-242, 2011, Springer Verlag.

[4] Lindgren H, Winnberg PJ, Winnberg P.: Domain Experts Tailoring Interaction to Users - an Evaluation Study. In P. Campos et al. (Eds.): INTERACT 2011, Part III, LNCS 6948, pp. 644-661, Springer 2011.

[5] Surie D, Pederson T, Lagriffoul F, Janlert LE, Sjölie D (2007) Activity Recognition using an Egocentric Perspective of Everyday Objects. In proc. of the IFIP International Conference on Ubiquitous Intelligence and Computing LNCS 4611:246-257 Springer

[6] Surie D, Jäckel F, Janlert LE, Pederson T (2010) Situative Space Tracking within Smart Environments. In proc. of the 6th International Conference on Intelligent Environments, IEEE Computer Society Press 152-157

[7] Surie D, Pederson T, Janlert LE (2010) The easy ADL home: A physical-virtual approach to domestic living. J of Ambient Intelligence and Smart Environments 2-3:287-310

[8] Nilsson I, Fisher AG (2006) Evaluating leisure activities in the oldest old. Scand J of Occupational Ther 13:31-37

[9] Ejlersen Wæhrens E (2010) Measuring quality of occupational performance based on self-report and observation : development and validation of instruments to evaluate ADL task performance. Umeå University, Doctoral thesis

[10] Chang, Y., Lim, Y., Stolterman, E.: Personas: From Theory to Practice. ACM. NordiCHI 2008, October 20-22, 2008.

[11] Campos, J., Paiva, A.: A Personal Approach: The Persona Technique in a Companion's Design Lifecycle. P. Campos et al. (Eds): INTERACT 2011, Part III, LNCS 6948, pp. 73-90, 2011.

[12] Carroll, JM. Scenario-Based Design. John Wiley, 1995

[13] Annicchiarico, R., Campana, F., Federici, A., Barrué, C., Cortés, U., Villar, A., Caltagirone, C.: Using Scenarios to Draft the Support of Intelligent Tools for Frail Elders in the SHARE-it Approach. In IWANN (1)(2009) 635-641

[14] Lindgren, H., Nilsson, I.: Towards User-Authored Agent Dialogues for Assessment in Personalised Ambient Assisted Living. Submitted.

[15] Lindgren, H.: Towards personalized decision support in the dementia domain based on clinical practice guidelines. User Modeling and User-Adapted Interaction 21(4):377-406 (2011)

[16] Lindgren H, Eriksson S. (2010) Sociotechnical Integration of Decision Support in the Dementia Domain. Stud Health Technol Inform 157;79-84, IOS Press

[17] Lindgren, H. Integrating Clinical Decision Support System Development into a Development Process of Clinical Practice - Experiences from Dementia Care. M. Peleg, N. Lavrac, and C. Combi (Eds.): AIME 2011, LNAI 6747, pp. 129-138, Springer 2011.

[18] Lindgren H.: Conceptual Model of Activity as Tool for Developing a Dementia Care Support System. In: Ackerman M, Dieng-Kuntz R, Simone C, Wulf V. (Eds.) Knowledge Management in Action (KMIA2008). IFIP 270, pp. 97-109, Springer Boston.

[19] American Psychiatric Association: Diagnostic and Statistical Manual of Mental Disorders, 4th edn., text revision (DSM-IV-TR). American Psychiatric Association (1994)

[20] Chesñevar C, McGinnis J, Modgil S, Rahwan I, Reed C, Simari G, South M, Vreeswijk G, Willmott S (2006) Towards an Argument Interchange Format. The Knowledge Engineering Review 21(4):293-316

[21] Lindgren H. Towards Context-Based Inquiry Dialogues for Personalized Interaction. In: Yves Demazeau, Michal Pechoucek, Juan M. Corchado and Javier Bajo (Eds.) PAAMS 2011. Advances in Intelligent and Soft Computing 88, pp. 151-161, Springer 2011, ISBN 978-3-642-19874-8

[22] Black E, Hunter A (2009) An inquiry dialogue system. Autonomous Agents and Multi-Agent Systems 19(2):173-209

[23] Walton DN, Krabbe ECW (1995) Commitment in dialogue: Basic concepts of interpersonal reasoning. SUNY Press

[24] García AJ, Simari GR (2004) Defeasible logic programming an argumentative approach. Theory and Practice of Logic Programming 4(12):95138

[25] Bellifemine, FB., Caire, G., Greenwood, D.: Developing Multi-Agent Systems with JADE. (Wiley Series in Agent Technology). John Wiley & Sons, 2007.

[26] Reed, C., Walton, D.: Towards a Formal and Implemented Model of Argumentation Schemes in Agent Communication. Journal Autonomous Agents and Multi-Agent Systems, Volume 11 Issue 2; 19-30, September 2005.

[27] I. Gómez-Sebastiá, J. Carlos Nieves (2010). WizArg: Visual Argumentation Framework Solving Wizard; In the Proceedings of the 13th International Conference of the Catalan Association of Artificial Intelligence. Pages 249-258, October, 2010, Tarragona, Spain.

[28] L. Riley, K. Atkinson, T. Payne and E. Black (2011): An implemented dialogue system for inquiry and persuasion. In: Proceedings of the First International Workshop on Theory and Applications of Formal Argumentation (TAFA 2011), Barcelona, Spain.

[29] Cortés U, Annicchiarico R, Urdiales C, Barrué C, Martínez A, Villar A; Agent Technology and e-Health, Whitestein Series in Software Technologies and Automatic Computing. pp:117-140

[30] Tolchinsky P, Modgil S, Cortés U (2006) Argument schemes and critical questions for heterogeneous agents to argue over the viability of a human organ. In AAAI Spring Symposium Series; Argumentation for Consumers of Healthcare 377-384

[31] Richard P. Cooper, J. Fox, D. W. Glasspool, P. Yule; Modelling High-level Cognitive Processes; May 2002; Publisher: Taylor & Francis, Inc.

A Qualitative Medical Diagnosis Approach based on Observer and Validating Agents

Juan Carlos Nieves and Helena Lindgren
Department of Computing Science
Umeå University
SE-901 87, Umeå, Sweden
Email: {jcnieves,helena}@cs.umu.se

Ulises Cortés
Universitat Politècnica de Catalunya
Software Department
c/Jordi Girona 1-3, E08034, Barcelona, Spain
Email: ia@lsi.upc.edu

ABSTRACT

In this paper we introduce a multi-agent approach designed to deal with medical qualitative diagnosis. This approach is based on BDI-agents which are called Observer and Validating agents. The Observer agents are supported by a deductive inference process and the Validating agents are supported by an abductive inference process. The knowledge bases of these agents are captured by a class of possibilistic logic programs. Hence these agents are able to deal with qualitative information. In order to manage an interaction between Observer and Validating agents, a deliberation dialogue process is defined. The approach is illustrated by real scenarios from diagnosing dementia disease.

1. INTRODUCTION

Qualitative information is a common ingredient which has to be dealt with in many real application domains. A clear example of these domains is the medical knowledge. Due to the ambiguities in and incompleteness of a medical knowledge domain, the clinical guidelines often apply expressions that mirror the status of the knowledge (*e.g.*, *possible*, *probable*, *supporting*, *unlikely*, *etc.*). Even in the case when a *medical guideline* (GL) uses apparently firm statements of presence or absence, it is commonly known (in this case as *tacit* knowledge) that assessments cannot be necessarily true. For instance, the major guideline for assessing mental conditions [1] states that Alzheimer's disease is present based on a set of observations, but only if all other medical conditions potentially causing the observed cognitive deficits are excluded. In an epistemic perspective, and in practice, it is impossible to exclude all other potential causes with certainty, meaning that Alzheimer's disease will never be assessed, and the guideline would be useless. Nevertheless, Alzheimer's disease is known to cause about 70 % of all cases of dementia, which allows also a highly unskilled physician to assess some dementia cases correctly, even if he does not know about any other condition causing dementia. On the other hand, considering that Alzheimer's disease is a fatal condition, typically leading to death within 5-8 years after receiving the diagnosis, the individuals who actually have dementias that can be *cured*, should not have to suffer from a misdiagnosed Alzheimer's disease. In fact, studies using autopsy (after death) as gold standard for dementia diagnosis have shown that only 49 % receives a correct diagnosis when diagnosed by experts, and in 37 % of the cases the diagnosis was changed completely [4].

Providing computerized systems for supporting the diagnostic process is a potential way to increase knowledge, quality and correctness in assessments, *e.g.*[9, 8].

In the literature, one can find different approaches for encoding qualitative information [11, 15, 16]. A common strategy for capturing qualitative information is by using non-numerical values.

Possibilistic reasoning has shown to be a suitable approach for dealing with qualitative reasoning [15]. In particular, this feature is based on the fact that the possibilistic values of a possibilistic knowledge can be non-numerical values which capture the uncertainty of a knowledge base. In the context of logic programming, there is an extension of answer set programming (ASP)[2] which combine possibilistic logic with a non-monotonic inference [12]. This extension of ASP is able to capture non-numerical values.

Against this background, in this paper we introduce a multi-agent approach for dealing with qualitative medical diagnosis. This approach is based on two kinds of BDI-agents:

- Observer agents: Observer agents are agents which take as input a set of observations from a patient and suggest a potential disease (condition). These agents are provided by a possibilistic knowledge base and a deductive reasoning method.
- Validating agents: Validating agents are agents which take as input a set of hypothesis which usually are the potential diseases which justify a set of observations. These agents are provided by a possibilistic knowledge base and an abductive reasoning method.

In order to manage a deliberation process between Observer and Validating agents, a deliberation dialogue process is defined.

The rest of the paper is divided as follows: in Section 2, a short motivation of the introduced qualitative medical diagnosis is presented. In Section 3, some basic concepts about possibilistic logic programs are presented. In Section 4, the formal definitions of the Observer and Validating agents are presented. In Section 5, a dialogue system is defined for managing a deliberation process between Observer and Validating agents. In Section 6, a short discussion *w.r.t.* related

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), June, 4–8, 2012, Valencia, Spain.

work is presented. In the last section, an outline of conclusions and future work are presented.

2. MOTIVATION

In this section, we discuss some general ideas which motivate our qualitative medical diagnosis approach.

Deductive reasoning can be seen as a basic form of medical diagnostic reasoning, forming decisions about conditions based on observations, typically as a multi-step process of refinement (*e.g.*, on basis of the set of observations $o_1, \ldots, o_n$, c can be assessed according to a knowledge base, *i.e.*, a formal interpretation of a guideline). To make the formalization of guidelines simple, the approach can be applied as in the exemplified clinical guideline [1]; assuming that everybody will know that assertions are not actually true in an epistemic perspective but represent the best decision at hand, knowing that this patient may be one of several exceptions. However, methods that can capture the complexity and defeasible characteristic of the generated knowledge in a patient case are becoming increasingly attractive, to provide appropriate support in domains pervaded with uncertainty. Since the ultimate purpose of knowledge-based diagnostic support is to educate the less skilled physician, the vagueness and incompleteness inherited from the evidence-based medical literature should be as explicit as possible. This enables also the provision of legitimate reasons for deviations from clinical guidelines in exceptional cases, which is highly important when evaluating the outcome of care.

Abductive diagnostic reasoning may also be seen as a form of diagnostic reasoning, where hypothesis generation and evaluation is included (*e.g.*, the set of observations $o_1, \ldots, o_n$, can be explained as being caused by c, considering c as a hypothesis, analyzed together with a knowledge base, the formal interpretation of a guideline) [11]. However, interpreting abductive diagnostic reasoning in this way, the knowledge base needs to contain information about diseases causing observed symptoms (causal information) in order to be able to explain the observations. Again, such statements need to capture the uncertainty of the knowledge domain.

In clinical practice, a combination of deductive and abductive reasoning is more applicable and could be described as a *hypothetico-deductive* approach [11]. A hypothetico- deductive reasoning method includes a hypothesis generation and evaluation procedure that resembles the medical diagnostic reasoning as it is done by medical professionals [14]. The algorithm can be summarized as follows:

1. Gather data through observations
2. Formulate hypothetical explanation
3. Deduce a consequence of explanation, predict, formulate experiment (test hypothesis)
4. Wait for corroboration:
   - **a** If corroboration, go to 3.
   - **b** If not corroboration, go to 2.

Step 3 differs from the abductive method, in that it involves decisions about what actions to take, e.g., do supplementary assessments to verify a hypothesis (filling in missing information or adding information to create a stronger case).

In the following sections, a multi-agent approach will be presented which try to follow the ideas of the *hypothetico-deductive* approach. In this setting, two kinds of agents are defined: the Observer and Validating agents.

3. BACKGROUND

In this section, we introduce some basic concepts of logic programs in the context of Possibilistic Logic Programming, for more details see [12].

We start introducing some concepts for standard disjunctive logic programs.

3.1 Non-Possibilistic Logic Programs

The language of a propositional logic has an alphabet consisting of

(i) proposition symbols: $p_0, p_1, ...$

(ii) connectives : $\vee, \wedge, \leftarrow, \neg, \ not, \bot$

(iii) auxiliary symbols : (,).

where $\vee, \wedge, \leftarrow$ are 2-place connectives, $\neg$, *not* are 1-place connective and $\bot$ is 0-place connective. The proposition symbols, $\bot$, and propositional symbols of the form $\neg p_i$ $(i \geq 0)$ stand for the indecomposable propositions, which we call *atoms*, or *atomic propositions*. The negation sign $\neg$ is regarded as the so called *strong negation* by the literature in Answer Set Programming and the negation *not* as the *negation as failure*. A literal is an atom, a, or the negation of an atom *not* a. Given a set of atoms $\{a_1, ..., a_n\}$, we write *not* $\{a_1, ..., a_n\}$ to denote the set of literals $\{not\ a_1, ..., not\ a_n\}$.

An extended disjunctive clause, C, is denoted:

$$a_1 \vee \ldots \vee a_m \leftarrow a_1, \ldots, a_j, not\ a_{j+1}, \ldots, not\ a_n$$

where $m \geq 0$, $n \geq 0$, each a_i is an atom. When $n = 0$ and $m > 0$ the clause is an abbreviation of $a_1 \vee \ldots \vee a_m$. When $m = 0$ the clause is an abbreviation of $\bot \leftarrow a_1, \ldots, a_n$ such that $\bot$ is the proposition symbol that always evaluates to false. Clauses of this form are called constraints (the rest, non-constraint clauses). An extended disjunctive program P is a finite set of extended disjunctive clauses. By $\mathcal{L}_P$, we denote the set of atoms in the language of P.

We denote an extended disjunctive clause C by $\mathcal{A} \leftarrow \mathcal{B}^+,\ not\ \mathcal{B}^-$, where $\mathcal{A}$ contains all the head atoms, $\mathcal{B}^+$ contains all the positive body atoms and $\mathcal{B}^-$ contains all the negative body atoms. When $\mathcal{B}^- = \emptyset$, the clause is called positive disjunctive clause. A set of positive disjunctive clauses is called a positive disjunctive logic program. When $\mathcal{A}$ is a singleton set, the clause can be regarded as a normal clause. A normal logic program is a finite set of normal clauses. Finally, when $\mathcal{A}$ is a singleton set and $\mathcal{B}^- = \emptyset$, the clause can be also regarded as a definite clause. A finite set of definite clauses is called a definite logic program.

The semantics for the disjunctive logic programs will be manage by the so called answer set semantics [6]. It is defined as follows: Let P be any extended disjunctive program. For any set $S \subseteq \mathcal{L}_P$, let P^S be the positive program obtained from P by deleting

(i) each rule that has a formula *not* a in its body with $a \in S$, and then

(ii) all formulae of the form *not* a in the bodies of the remaining rules.

Clearly P^S does not contain *not* (this means that P^S is either a positive disjunctive logic program or a definite logic program), hence S is an answer set of P if and only if S is a minimal model of P^S.

Now that we have already defined the syntaxis and semantics for disjunctive logic programs, the syntaxis and semantics for possibilistic logic programs will be defined.

3.2 Possibilistic Logic Programs

We want to point out that in the whole document only finite lattices are considered. This assumption was made based on the recognition that in real applications we will rarely have an infinite set of labels for expressing the incomplete state of a knowledge base.

A *possibilistic atom* is a pair $p = (a, q) \in \mathcal{A} \times \mathcal{Q}$, in which $\mathcal{A}$ is a finite set of atoms and $(\mathcal{Q}, \leq)$ is a lattice. The projection $*$ to a possibilistic atom p is defined as follows: $p^* = a$. Also given a set of possibilistic atoms S, $*$ over S is defined as follows: $S^* = \{p^* | p \in S\}$.

Let $(\mathcal{Q}, \leq)$ be a lattice. A possibilistic disjunctive clause R is of the form:

$$\alpha : a_1 \vee \ldots \vee a_m \leftarrow a_{m+1} \wedge \cdots \wedge a_j \wedge not\ a_{j+1} \wedge \cdots \wedge not\ a_n$$

in which $\alpha \in \mathcal{Q}$ and each $a_i (1 \leq i \leq n)$ is an atom. Sometimes a possibilistic disjunctive clause R is denoted by $\alpha : \mathcal{A} \leftarrow \mathcal{B}^+ \wedge\ not\ \mathcal{B}^-$.

The projection $*$ for a possibilistic clause is $R^* = \mathcal{A} \leftarrow \mathcal{B}^+ \wedge\ not\ \mathcal{B}^-$. On the other hand, the projection n for a possibilistic clause is $n(R) = \alpha$. This projection denotes the degree of necessity captured by the certainty level of the information described by R.

> "α is not a probability (like it is in probability theory), but it induces a *certainty* (or *confidence*) scale. This value is determined by the expert providing the knowledge base"

A possibilistic constraint C is of the form:

$$\top_{\mathcal{Q}} :\ \leftarrow \mathcal{B}^+ \wedge\ not\ \mathcal{B}^-$$

in which $\top_{\mathcal{Q}}$ is the top of the lattice $(\mathcal{Q}, \leq)$. The projection $*$ for a possibilistic constraint C is: $C^* = \leftarrow \mathcal{B}^+ \wedge\ not\ \mathcal{B}^-$.

A possibilistic disjunctive logic program P is a tuple of the form $\langle (\mathcal{Q}, \leq), N \rangle$, in which N is a finite set of possibilistic disjunctive clauses and possibilistic constraints. The generalization of $*$ over P is as follows: $P^* = \{r^* | r \in N\}$. Notice that P^* is an extended disjunctive program. When P^* is a normal program, P is called a possibilistic normal program. Also, when P^* is a positive disjunctive program, P is called a possibilistic positive logic program and so on. A given set of possibilistic disjunctive clauses $\{\gamma, \ldots, \gamma\}$ is also represented as $\{\gamma; \ldots; \gamma\}$.

We illustrate a possibilistic disjunctive logic program with an example from the dementia domain (simplified due to space reasons). A summary of the clinical guidelines which are used in the dementia example given here can be found in [13] and includes [1]. We use the following abbreviations:

AD	=	*Alzheimer's disease*
DLB	=	*Lewy body type of dementia*
VaD	=	*Vascular dementia*
$epiMem$	=	*Episodic memory dysfunction*
$fluctCog$	=	*Fluctuating cognition*
fn	=	*Focal neurological signs*
$prog$	=	*Progressive course*
$radVasc$	=	*Radiology exam shows vascular signs*
$slow$	=	*Slow, gradual onset*
$extraPyr$	=	*Extrapyramidal symptoms*
$visHall$	=	*Visual hallucinations*

We extract the following labels describing different levels of uncertainty of assessments from the clinical guidelines: $\mathcal{Q} := \{confirmed, probable, possible, plausible, supported, open\}$. To describe their relationships, let $<$ be a partial order such that the following set of relations holds:

$\{confirmed > probable, probable > possible, confirmed > plausible, plausible > supported, possible > supported, supported > open\}$

The graphic representation of this lattice is presented in Figure 1.

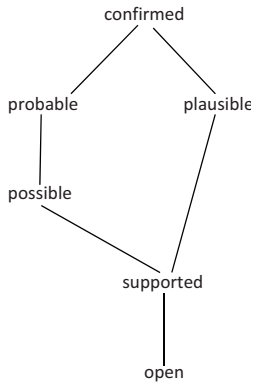

Figure 1: A Lattice

Given $x, y \in \mathcal{Q}$, the relation $x > y$ means that y is less certain than x.

Example 1. *The following clauses are included in our possibilistic logic program:*

1. **probable***:* $VaD \leftarrow fn \wedge radVasc \wedge\ not\ (AD \vee DLB)$
2. **probable***:* $DLB \leftarrow extraPyr \wedge visHall \wedge not\ fn$
3. **probable***:* $DLB \leftarrow fluctCog \wedge visHall \wedge not\ fn$
4. **probable***:* $DLB \leftarrow fluctCog \wedge extraPyr \wedge not\ fn$
5. **probable***:* $VaD \leftarrow fn \wedge radVasc$
6. **probable***:* $DLB \leftarrow extraPyr \wedge fluctCog$

7. **possible**: $VaD \leftarrow fn \wedge fluctCog$

8. **possible**: $DLB \leftarrow fn \wedge fluctCog$

9. **possible**: $VaD \leftarrow fn \wedge slow \wedge prog \wedge epiMem$

10. **possible**: $AD \leftarrow fn \wedge slow \wedge prog \wedge epiMem$

11. **possible**: $VaD \leftarrow radVasc \wedge slow \wedge prog \wedge epiMem$

12. **possible**: $AD \leftarrow radVasc \wedge slow \wedge prog \wedge epiMem$

13. **possible**: $DLB \leftarrow fluctCog \wedge slow \wedge prog \wedge epiMem$

14. **possible**: $AD \leftarrow fluctCog \wedge slow \wedge prog \wedge epiMem$

15. **possible**: $DLB \leftarrow extraPyr \wedge slow \wedge prog \wedge epiMem$

16. **possible**: $AD \leftarrow extraPyr \wedge slow \wedge prog \wedge epiMem$

17. **possible**: $DLB \leftarrow visHall \wedge slow \wedge prog \wedge epiMem$

18. **possible**: $AD \leftarrow visHall \wedge slow \wedge prog \wedge epiMem$

19. **possible**: $DLB \leftarrow fluctCog$

20. **possible**: $DLB \leftarrow visHall$

21. **possible**: $DLB \leftarrow extraPyr$

22. **possible**: $VaD \leftarrow fn$

23. **possible**: $VaD \leftarrow radVasc$

24. **supported**: $VaD \leftarrow fluctCog$

25. **plausible**: $VaD \leftarrow fn$

Similar to the definition of answer set semantics [5], the possibilistic answer set semantics is defined in terms of a syntactic reduction.

Definition 1 (Reduction P_M). *[12] Let $P = \langle(\mathcal{Q}, \leq), N\rangle$ be a possibilistic disjunctive logic program, M be a set of atoms. P reduced by M is the positive possibilistic disjunctive logic program:*

$$P_M := \{(n(r) : \mathcal{A} \cap M \leftarrow \mathcal{B}^+) | r \in N, \mathcal{A} \cap M \neq \emptyset, \mathcal{B}^- \cap M = \emptyset, \mathcal{B}^+ \subseteq M\}$$

in which r^* is of the form $\mathcal{A} \leftarrow \mathcal{B}^+$, not $\mathcal{B}^-$.

One can observe that the reduced program P_M is a possibilistic positive logic program.

Given a possibilistic positive logic program, one can apply a simple possibilistic version of the generalized partial evaluation principle.

Definition 2 (Grade-GPPE (G-GPPE)). *[12] Let r_1 be a possibilistic clause of the form $\alpha : \mathcal{A} \leftarrow \mathcal{B}^+ \cup \{B\}$ and r_2 a possibilistic clause of the form $\alpha_1 : \mathcal{A}_1$ such that $B \in \mathcal{A}_1$ and $B \notin B^+$, then*

$$\textit{G-GPPE}(r_1, r_2) = (\mathcal{GLB}(\{\alpha, \alpha_1\}) : \mathcal{A} \cup (\mathcal{A}_1 \setminus \{B\}) \leftarrow \mathcal{B}^+)$$

Observe that basically G-GPPE is adding new possibilistic positive clauses to the given program.

By considering G-GPPE one can define the operator $\mathcal{T}$ as follows:

Definition 3. *Let P be a possibilistic positive logic program. The operator $\mathcal{T}$ is defined as follows:*

$$\mathcal{T}(P) := P \cup \{\textit{G-GPPE}(r_1, r_2) | r_1, r_2 \in P\}$$

An interesting thing of this operator $\mathcal{T}$ is that it always reaches a fix-point [12]. Hence, given a possibilistic positive logic program P, $\Pi(P)$ denotes the fix-point of the operator $\mathcal{T}$ *w.r.t.* P.

Since $\Pi(P)$ added new possibilistic positive clauses to P which could have an empty body, in the following definition the function $Sem_{min}(P)$ is introduced. This function returns the set of possibilistic atoms from $\Pi(P)$ which have an empty body.

Definition 4. *[12] Let P be a possibilistic logic program and $Facts(P, a) := \{(\alpha : a) | (\alpha : a) \in P\}$. $Sem_{min}(P) := \{(x, \alpha) | Facts(P, x) \neq \emptyset$ and $\alpha := \mathcal{LUB}(\{n(r) | r \in Facts(P, x)\})\}$ in which $x \in \mathcal{L}_P$.*

By considering $Sem_{min}(P)$ and the fix-point $\Pi(P)$, the possibilistic answer set semantics is defined as follows:

Definition 5. *[12] Let P be a possibilistic disjunctive logic program and M be a set of possibilistic atoms such that M^* is an answer set of P^*. M is a possibilistic answer set of P if and only if $M = Sem_{min}(\Pi(P_{M^*}))$.*

In order to illustrate the definition of answer sets for possibilistic logic programs, let us consider a subset of possibilistic rules which were introduced in Example 1.

Example 2. *Let $P = \langle(\mathcal{Q}, \leq), N\rangle$ be a possibilistic logic programs in which $(\mathcal{Q}, \leq)$ is the lattice introduced in Example 1 and N is the following set of possibilistic rules:*

$confirmed:$	$fn \leftarrow \top$
$confirmed:$	$radVasc \leftarrow \top$
$confirmed:$	$extraPyr \leftarrow \top$
$confirmed:$	$fluctCog \leftarrow \top$
$probable:$	$VaD \leftarrow fn \wedge radVasc \wedge$
$probable:$	$DLB \leftarrow extraPyr \wedge fluctCog$
$possible:$	$DLB \leftarrow extraPyr$

In order to infer the answer sets of P, the first step is to find, the answer set of P^*. It is not hard to see that P^* has only one answer set which is $M = \{fn, radVasc, extraPyr, fluctCog, DLB, VaD\}$. Now, since P has no rules with negative literal, P_M is the same to P.

One can see that the subset of possibilistic rules from $\Pi(P)$ which have an empty body is:

$confirmed:$	$fn \leftarrow \top$
$confirmed:$	$radVasc \leftarrow \top$
$confirmed:$	$extraPyr \leftarrow \top$
$confirmed:$	$fluctCog \leftarrow \top$
$probable:$	$VaD \leftarrow \top$
$probable:$	$DLB \leftarrow \top$
$possible:$	$DLB \leftarrow \top$

Hence, $Sem_{min}(\Pi(P_{M^*})) = \{(fn, confirmed), (radVasc, confirmed), (extraPyr, confirmed), (fluctCog, confirmed), (DLB, probable), (VaD, probable)\}$ Hence, we say that $Sem_{min}(\Pi(P_{M^*}))$ is an answer set of P.

In the following section, the inference of the possibilistic semantics will be used for defining the inference mechanisms of the Observer and Validating agents. Therefore, in the following definition a particular notation of the inference of the possibilistic answer set semantics is introduced.

DEFINITION 6. *Let $P = \langle(\mathcal{Q}, \leq), N\rangle$ be a possibilistic logic program and S be a set of possibilistic atoms. $P \models S$ holds if there exists a possibilistic answer set M of P such that $S \subseteq M$.*

4. OBSERVER AND VALIDATING AGENTS

In this section, the ideas of Observer and Validating agents will be presented. The general idea is that both Observer and Validating agents will collaborate in order to improve the quality of a potential diagnosis. Both Observer and Validating agents will be provided with a possibilistic knowledge base.

4.1 Observer agents

An Observer agent is based on a *possibilistic deductive reasoning method*. In particular this deductive method is instantiated with the possibilistic logic programming semantics.

DEFINITION 7. *A Observer agent A_o is a tuple $\langle\Sigma, O, B\rangle$ in which Σ is a possibilistic logic program, O is a set of possibilistic atoms which are called observations, B is a set of possibilistic atoms which are called beliefs and the following condition holds: $\Sigma \cup O \models B$.*

Take into account that an Observer agent is basically an agent which has a possibilistic knowledge base. It can get a set of observations from the world and by using a deductive inference it gets a view of the world which is captured by the set of beliefs of the world.

EXAMPLE 3. *Let $A_o^1 = \langle\Sigma_1, O_1, B_1\rangle$ be an Observer agent such that O_1 is an empty set and Σ_1 is the following possibilistic program:*

$$\begin{array}{ll} probable: & VaD \leftarrow fn \wedge radVasc \\ probable: & DLB \leftarrow extraPyr \wedge fluctCog \\ possible: & DLB \leftarrow extraPyr \end{array}$$

One can see that given that O_1 is empty; hence, the set of beliefs B_1 of A_O^1 is empty.

Observe that the knowledge base of A_o^1 basically is capturing some knowledge for diagnosing Vascular dementia *(VaD) and* Lewy body type of dementia *(DLB). Now, let us suppose that A_O^1 gets the following set of observations O_1' from a patient PP:*

$$\begin{array}{ll} confirmed: & fn \leftarrow \top \\ confirmed: & radVasc \leftarrow \top \\ confirmed: & extraPyr \leftarrow \top \\ confirmed: & fluctCog \leftarrow \top \end{array}$$

In this case, one can see that the set of beliefs of A_o^1 will be the answer sets of the program $\Sigma_1 \cup O_1'$. As we saw in Example 2, this program has only one answer set which will be denoted by M. Hence M will be the set of beliefs of A_o^1. One can see that $\{(DLB, probable), (VaD, probable)\} \subseteq M$. This means that A_O^1 can believe that it is probable *that PP could have either* Vascular dementia *(VaD) or* Lewy body type of dementia*. At this state of the diagnosis, these potential diseases are only considered as potential hypothesis which could explain the state of PP. In the following section, the Validating agents will apply an abductive inference for validating the potential diagnosis.*

4.2 Validating Agents

In this subsection, the idea of Validating agents will be introduced. A Validating agent will be a specialized agent in a particular domain which will validate a potential diagnosis by using an abductive inference approach. Indeed, a Validating agent will take as an input the potential beliefs from an Observer agent.

In order to define the Validating agents, the concepts of a possibilistic abductive diagnostic problem and a possibilistic diagnosis will be defined.

Usually a (technical) diagnostic problem consists of a description of a technical system to be diagnosed, observations of the actual state of the system, and the potential reasons for effects. Hence, *a possibilistic abductive diagnostic problem* for this purpose is defined as follows:

DEFINITION 8. *A possibilistic abductive diagnostic problem (PADP) is a triple $\langle H, T_P = \langle(\mathcal{Q}, \leq), N\rangle, O\rangle$ in which:*

- *H is a set of possibilistic atoms such that $\{\alpha | (a, \alpha) \in H\} \subseteq \mathcal{Q}$. H is called the set of hypotheses.*
- *T_P is a possibilistic logic program which is called a possibilistic theory.*
- *O is a set of atoms which are called observations.*

Observe that the set of observations is a set of non-possibilistic atoms. It is expected that the possibilistic theory suggests an uncertain degree to each element of the observations.

By considering the semantics for possibilistic logic programs, a possibilistic diagnosis of a possibilistic abductive diagnostic problem is defined as follows:

DEFINITION 9. *Let $\langle H, T_P = \langle(\mathcal{Q}, \leq), N\rangle, O\rangle$ be a possibilistic abductive diagnostic problem. A possibilistic diagnosis is a tuple $\langle H', O_P\rangle$ such that $H' \subseteq H$, $T_P \cup \{\alpha : h \leftarrow \top | (h, \alpha) \in H'\} \models O_P$ and $(O_P)^* = O$.*

Observe that a possibilistic diagnosis not only gives evidence for explaining a set of observations, but also it identifies an uncertain degree for each observation. Given that a possibilistic abductive diagnostic problem can have different diagnoses, the idea of a minimal possibilistic diagnosis is defined as follows:

DEFINITION 10. *Let $PADP = \langle H, T_P, O\rangle$ be a possibilistic abductive diagnostic problem. A possibilistic diagnosis $D_1 = \langle H_1', O_P 1\rangle$ of $PADP$ is a minimal possibilistic diagnosis if there is not a diagnosis $D_2 = \langle H_2', O_P 2\rangle$ of $PADP$ such that $(H_2')^* \subset (H_1')^*$.*

By having in mind, the concepts of a possibilistic abductive diagnostic problem and a possibilistic diagnosis, *a Validating agent* is defined as follows:

DEFINITION 11. *A Validating agent A_v is a tuple of the form $\langle PADP, \mathcal{D}\rangle$ in which $PADP = \langle H, T_P = \langle(\mathcal{Q}, \leq), N\rangle, O\rangle$ is a possibilistic abductive diagnostic problem and $\mathcal{D}$ is a set of diagnoses w.r.t. $PADP$.*

Consider again the example from the dementia domain. If we want to accomplish a hypothetico-deductive reasoning process, we may proceed through the first two steps using *an Observer agent* as described in previous subsection to generate a set of hypotheses. In the third step, the hypothesis is evaluated and *possibly challenged*. In order to enrich our diagnosis, *a Validating agent* may be used for deciding upon what to observe, *e.g.*, if we have deduced the possible coexistence of AD and VaD, we may use the abductive reasoning inference to determine what features to investigate in order to create a stronger case for the hypothesis.

In order to capture the possibilistic knowledge base of a Validating agent, we re-interpret the clinical guidelines in order to explore what information they give on causality, *i.e.*, what can we expect to observe in an individual with a certain disease. Ideally, we would use reliable probability measures, but since these are not available or not stable over the disease progression, we use interpretations of the expressions indicating likelihood. The intuitive interpretation will be:

> "It is *always/likely/typically/possibly/rare* that the diagnosis D causes/explains the phenomenon O observable at some point during the disease progression".

Hence, we create another lattice that captures this causality.

EXAMPLE 4. *Our second lattice consists of the following:* $\mathcal{Q} := \{always, likely, typically, possibly, rare\}$. *Their relationships are defined as follows:* $\{always > likely$, $likely > typically$, $likely > possibly$, $possibly > rare$, $typically > rare\}$. *The graphic representation of this lattice is presented in Figure 2*

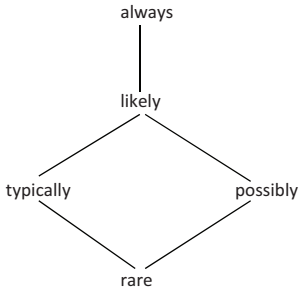

Figure 2: A Lattice

Let $\Sigma_2 = \langle \mathcal{Q}, N \rangle$ *such that* N *is the following set of clauses:*

1. **likely**: $extraPyr \leftarrow DLB$
2. **likely**: $fluctCog \leftarrow DLB$
3. **likely**: $visHall \leftarrow DLB$
4. **always**: $fn \leftarrow VaD$
5. **likely**: $radVasc \leftarrow VaD$
6. **typically**: $fluctCog \leftarrow VaD$
7. **always**: $epiMem \leftarrow AD$
8. **always**: $slow \leftarrow AD$
9. **always**: $prog \leftarrow AD$
10. **possibly**: $extraPyr \leftarrow AD$

The expression that DLB is *likely* to cause extrapyramidal symptoms is based on the clinical knowledge that such symptoms have been observed in up to 70 % of DLB cases in EBM studies, and have been identified as one of three core symptoms for diagnosis [7]. The expression that AD *possibly* causes extrapyramidal symptoms is based on that such symptoms have been observed in up to 30 % of AD cases, where DLB has been excluded with reliable methods.

EXAMPLE 5. *Let us consider Example 3. According to agent* A^1_O*, there are observations which support that it is possible that the given patient could have either vascular dementia (VaD) or lewy body type dementia (DLB). This means that* A^1_O *suggests the following set of hypothesis:*

$$Hyphotheses = \{(DLB, probable), (VaD, probable)\}$$

However, these are still too uncertain to be satisfactory for committing to a final diagnosis. The question is: can we use a validating agent as a next step in the diagnosis to evaluate the hypothetical diagnoses, following the hypothetico-deductive approach and find out what to do as a next step in the assessment?

Let $PADP_1 = \langle H, T_P = \langle (\mathcal{Q}, \leq), N \rangle, O \rangle$ *be a possibilistic abductive diagnostic problem such that*

$H = \{(DLB, always), (VaD, always)\}$
$O = \{Obs = \{(fn, confirmed), (fluctCog, confirmed)\}^*$

and T_P *is the possibilistic programs introduced in Example 4.*

In order to identify the explanations (diagnoses) of $PADP_1$*, let*

$$H_1 = \{(DLB, always), (VaD, always)\}$$

$$H_2 = \{(DLB, always)\}$$

and $H_3 = \{(VaD, always)\}$. *The label* always *for potential hypotheses indicates that they always should be considered as potential explanations for, or causes of, a particular set of observations. In order to see if* H_1 *should be considered as an explanation (defines a possibilistic diagnosis) of* $PADP$*, let* P_{H_1} *be* T_P *union the following possibilistic rules:*

$always: \quad DLB \leftarrow \top$
$always: \quad VaD \leftarrow \top$

One can see that P_{H_1} *has a unique possible answer set:*
$M_{H_1} = \{(extraPyr, likely), (fluctCog, likely), (visHall, likely), (fn, always), (radVasc, likely), (fluctCog, typically), (DLB, always), (VaD, always)\}$.
Let $O_{H_1} = \{(fn, always), (fluctCog, typically)\}$.
Since $O_{H_1} \subseteq M_{H_1}$ *and* $O^*_{H_1} = O$, $\langle H_1, O_{H_1} \rangle$ *is a possibilistic diagnosis of* $PADP_1$.

Let us check if H_2 *defines a possibilistic diagnosis of* $PADP_1$. *Hence, let* P_{H_2} *be* T_P *union the following possibilistic rule:*

$always: \quad DLB \leftarrow \top$

P_{H_2} has a unique possible answer set:
$M_{H_2} = \{(extraPyr, likely), (fluctCog, likely), (visHall, likely), (DLB, always)\}$

Since $O \nsubseteq M^*_{H_2}$, H_2 does not define a possibilistic diagnosis of $PADP$.

Now let us check if H_3 defines a possibilistic diagnosis of $PADP$. So let P_{H_3} be T_P union the following possibilistic rule:

$always: \quad VaD \leftarrow \top$

P_{H_3} has a unique possible answer set $M_{H_3} = \{(fn, always), (radVasc, likely), (fluctCog, typically), (VaD, always)\}$. Let $O_{H_3} = \{(fn, always), (fluctCog, typically)\}$. Since $O_{H_3} \subseteq M_{H_3}$ and $O^*_{H_3} = O$, $\langle H_3, O_{H_3}\rangle$ is a possibilistic diagnosis of $PADP_1$.

So far, we have found two possibilistic diagnoses of $PADP$: $D = \langle H_1, O_{H_1}\rangle$ and $D' = \langle H_3, O_{H_3}\rangle$. Hence, a validating agent A^1_v can be instantiated in the following tuple:

$$A^1_v = \langle PADP_1, \{\langle H_1, O_{H_1}\rangle, \langle H_3, O_{H_3}\rangle\}\rangle$$

One can observe that A^1_v has a unique minimal possibilistic diagnosis w.r.t. $PADP_1$ which is $\langle H_3, O_{H_3}\rangle$. Hence, the Validating agent A^1_v can conclude VaD is the strongest candidate for explaining the available observations. However, since more information is needed, the results can also be used to identify the next step in the assessment process. The physician can be informed about which additional observations would strengthen the hypotheses by using M_{H_1} and M_{H_3}.

5. DELIBERATION DIAGNOSIS

So far, we have defined the inference processes of the Observer and Validating agents. Indeed, we have illustrated how an Observer agent can suggest a set of hypotheses which could diagnose a potential disease. On the other hand, we have illustrated that Validating agents can agree or disagree with respect to the diagnose of an Observer agent. However, some mechanisms for deliberating a potential diagnosis between Observer and Validating agents are required. To this end, we introduce a dialog system for managing a dialog deliberation process between Observer and Validating agents.

The basic piece in a deliberation dialog process will be a *move*. By following some ideas of [3], we consider three types of moves: *open*, *assert* and *close*.

DEFINITION 12. *Given an Observer agent A_o and a finite set of Validating agents $\mathcal{A}_v$, the following moves are defined:*

Open: *An open move is of the form $\langle A_o, open, \langle O, B\rangle\rangle$ and $A_o = \langle \Sigma, O, B\rangle$.*

Assert: *An assert move is of the form $\langle A_v, assert, \langle H, O\rangle\rangle$ in which $A_v \in \mathcal{A}_v$ and $A_v = \langle PADP, \mathcal{D}\rangle$ and $\langle H, O\rangle \in \mathcal{D}$*

close: *A close move is of the form $\langle A, close\rangle$ such that $A = A_o$.*

The set of moves introduced by Definition 12 will be denoted by $\mathcal{M}$. Given a set of Observer and Validating agents $\mathcal{A}$. A $Sender : \mathcal{M} \longmapsto \mathcal{A}$ is a function such that $Sender(m) = A$, $m \in \mathcal{M}$ and $A \in \mathcal{A}$.

A deliberation dialog is simply a finite sequence of moves. We assume that the first move will be done by an Observer agent. Like the approach presented in [3], a dialogue progresses over time; hence, each time point is denoted by a natural number.

DEFINITION 13. *Let A_o be an Observer agent and $\mathcal{A}_v$ be a finite set of Validating agents. A deliberation dialogue D^t is a finite sequence of moves $[m_0, \ldots, m_t]$ such that $t \in \mathbb{N}$ and the following conditions holds:*

1. *m_0 is a move of the form $\langle A_o, open, \langle O, B\rangle\rangle$*
2. *if $m_s = \langle A_v, assert, \langle H, O\rangle\rangle$ such that $A_v \in \mathcal{A}_v$,$(0 < s < r)$, then $m_{s-1} = \langle A_o, T, \langle O', B\rangle\rangle$ such that $T \in \{open, assert\}$ and $B \subseteq H$.*
3. *if $m_s = \langle A_o, assert, \langle O, B\rangle\rangle$ such that $(0 < s < r)$, then $m_{s-1} = \langle A_v, assert, \langle H, O'\rangle\rangle$ such that $A_v \in \mathcal{A}_v$ and $O^* \subseteq O'^*$.*

Given a deliberation dialogue D^t and a set of Validating agents $\mathcal{A}_v$, the set of active Validating agents *w.r.t.* D^t is $\mathcal{A}^{act}_v(D^t) = \{A_v | m = \langle A_v, assert, \langle H, O\rangle\rangle$ and $m \in D^t\}$ and the active Observer agent *w.r.t.* D^t is $A^{act}_o(D^t) = A_o$ such that $m_0 = \langle A_o, Open, \langle O, B\rangle\rangle$ and $m_o \in D_t$. Observer that any deliberation dialogue has only one an active Observer agent and one or more Validating agents. This restriction of a deliberation dialogue is motivated by the fact that, in a deliberation dialogue, the Observer agent's diagnosis is discussed by several specialized Validating agents.

DEFINITION 14. *A deliberation dialogue $D^t = [m_0, \ldots, m_t]$ is close iff $m_t = \langle A, close\rangle$ and $Sender(m_t) = A^{act}_o(D^t)$.*

This definition suggests that a deliberation dialogue is closed only by the Observer agent that opened it. In order to illustrate these definition let us consider the following example.

EXAMPLE 6. *Let us continue with our running example which has been discussed in Example 3 and Example 5. Hence, according to Example 3, a possible instantiation of an Observer agent is:*

$$\begin{aligned} A^1_o = \quad & \langle \Sigma_1, \\ & O' = \{confirmed : fn \leftarrow \top; \\ & confirmed : radVasc \leftarrow \top; \\ & confirmed : extraPyr \leftarrow \top; \\ & confirmed : fluctCog \leftarrow \top\} \\ & B' = \{(DLB, probable), (VaD, probable)\}\rangle \end{aligned}$$

Therefore, one can say that according to its knowledge and the observations, A^1_o believes that it is probable *that a given patient could have either* Vascular dementia *(VaD) or* Lewy body type of dementia.

By considering Example 5, a possible instantiation of a Validating agent is:

$$A^1_v = \langle PADP_1, \{\langle H_1, O_{H_1}\rangle, \langle H_3, O_{H_3}\rangle\}\rangle$$

A possible deliberation dialogue could run as follows:

t	$move$
0	$\langle A_o^1, open, \langle O', B' \rangle \rangle$
1	$\langle A_v^1, assert, \langle H_1, 0_{H_1}, \rangle \rangle$
2	$\langle A_o^1, close \rangle$

In the first move, A_o^1 suggests a possible diagnosis which is part of its beliefs. In the second move, A_v^1 introduces an explanation of the diagnosis suggested by A_o^1. Given that A_o^1 has no new information, A_o^1 closes the deliberation dialogue.

At this state of our research, one can see a deliberation dialogue as a line dialogue in which an Observer agent suggests a potential diagnosis in the presence of a set of observations and a Validating agent suggests a potential explanation of the given diagnosis. If an Observer agent gets new information, it could suggest a new diagnosis and other Validating agent could explain the new diagnosis.

Let us observe that the current dialogue protocol does not allow a move of disagreement in the suggested diagnosis. In our future work, we will extend our protocol for supporting disagreements between the active participants of a deliberation dialogue.

6. DISCUSSION

In medicine the goal for diagnostic reasoning is assessing causes of observed conditions in order to make informed choices about treatment. The knowledge about causes of diseases is preferably created in randomized clinical trials, generating *evidence-based* medical (EBM) knowledge. This knowledge is based on probabilities, *e.g.*, if there is evidence that a certain disease is causing a given observed or measured phenomenon in a proportion of all cases of this disease. In addition, knowledge about the proportion of the manifested phenomenon in the total population is needed, including subjects not having the disease in order to assess the diagnostic value of the observation. If the observation has a high diagnostic value (*i.e.*, seen in a large proportion of cases with the disease and in a low proportion of cases without the disease) it is typically included in medical guidelines for diagnosis.

In practice, the diagnostic guidelines are interpretations of the EBM knowledge, aiming to overcome ambiguities and incompleteness of the available and evolving domain knowledge. One can view the evidence-based, statistically based, medical knowledge as being the generic knowledge about a medical domain, but it is often of a limited aid in assessing an individual's condition.

Diagnosis is not as much about what a clinician *knows* about a domain, but *knowing what to do*, and knowing *how to apply* the knowledge (*i.e.*, [14]). Diagnosis is primarily a problem-solving process. The skills to perform this differ among clinicians depending on education, training, experience (*i.e.*, level of expertise) and contextual factors such as workload and work content. It is shown that novice doctors tend to reason using a causal procedure (starting with a potential diagnosis and searching for evidence to prove this condition), following the EBM knowledge they have acquired during medical education. In addition, it has been shown that experienced physicians tend to use a causal reasoning when explaining assessments to *e.g.*, medical students [14]. However, the risk to miss important information is high when relying only on causal knowledge when conducting assessment. The diagnostic reasoning process applied by experts is described as a process where observations are typically collected without jumping to conclusions too early, in a process of creating the base for moving forward towards a diagnosis. Typically, at the point when hypotheses are formulated, the hypotheses finally selected are included with the set of experts' reasoning, which is not necessarily the case with novice doctors' reasoning [14].

Consequently, one desired property of a medical diagnostic support system is to be able to support the diversity of human reasoning in the diagnostic problem-solving process and potentially aid the transformation from novices' type of reasoning towards applying diagnostic reasoning rather than causal reasoning. Hence, by combining two kinds of reasoning (*deductive and abductive reasoning*), one can resemble when needed the reasoning type of novice and expert doctors. Let us remember that a combination of deductive and abductive reasoning could be described as a *hypothetico-deductive* approach (see Section 2).

Other essential properties are the capability to handle comorbidity manifested in patients, and be able to capture the incomplete, ambiguous and uncertain medical knowledge. Comorbidity may imply the expertise in to medical specialities. The modeling of this knowledge needs to be done by medical professionals, which requires formalisms that are intuitive and transparent to capture the model. To achieve this the Observer and Validating agents are provided with possibilistic knowledge bases. In particular, we have applied *possibilistic logic programs*. The possibilistic programs have the advantage of using ordered labels for capturing the uncertainty from medical knowledge.

7. CONCLUSIONS AND FUTURE WORK

In the medical context, we have argued that possibilistic logic programs define a rich approach for capturing real medical knowledge. Indeed, it seems that the introduced qualitative diagnosis approach has practical applications in the medical diagnosis since it combines different strategies in a diagnostic reasoning process in a similar way as the human approaches the task. In this way the human (*i.e.*, expert or novice clinician) may gain support tailored to his or her need in a collaborative and transparent reasoning and problem-solving process.

The consideration of intelligent systems which could suggest and/or validate a potential disease could improve the quality of a medical diagnosis which is done by a novice clinician. Hence, the consideration of intelligent systems which could follow the approach of Observer and Validating agents can impact in the early detection of mental diseases.

Our approach allows a first step towards the use of a *manageable* and sound formalism to ease the (medical) diagnosis procedure. It allows to represent several reasoning procedures and, if necessary, to combine those to allow stronger diagnosis procedures. Hence, it allows the *explanation* of the reasoning procedures.

In our future work, there are two main issues:

1. To improve the deliberation protocol for dealing with disagreements between the participants of a deliberation dialogue.

2. To evaluate in practical knowledge modeling and diagnostic situations involving medical professionals as part of the ACKTUS project [10].

Acknowledgement

This research is partly funded by VINNOVA (Sweden's innovation agency) and the Swedish Brain Power. This research is also partially funded by (CIP-ICT-PSP-2011-5 #297225) "Integrated prevention and Detection sOlutioNs Tailored to the population and Risk Factors associated with FALLs (I-DONT-FALL)".

8. REFERENCES

[1] American Psychiatric Association. *Diagnostic and Statistical Manual of Mental Disorders DSM-IV-TR Fourth Edition (Text Revision)*. Amer Psychiatric Pub, 4th edition, 2000.

[2] C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, Cambridge, 2003.

[3] E. Black and A. Hunter. An inquiry dialogue system. *Autonomous Agents and Multi-Agent Systems*, 19(2):173–209, 2009.

[4] H. Brunnström and E. Englund. Clinicopathological concordance in dementia diagnostics. *Am J Geriatr Psychiatry*, 17(8):664–70, 2009.

[5] M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In R. Kowalski and K. Bowen, editors, *5th Conference on Logic Programming*, pages 1070–1080. MIT Press, 1988.

[6] M. Gelfond and V. Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing*, 9:365–385, 1991.

[7] F. Geser, G. K. Wenning, W. Poewe, and I. McKeith. How to diagnose dementia with lewy bodies: state of the art. *Mov Disord*, 20 Suppl 12:S11–20, 2005.

[8] H. Lindgren. Limitations in physicians' knowledge when assessing dementia diseases - an evaluation study of a decision-support system. *Studies In Health Technology And Informatics*, 169:120–124, 2011.

[9] H. Lindgren. Towards personalized decision support in the dementia domain based on clinical practice guidelines. *User Model. User-Adapt. Interact.*, 21(4-5):377–406, 2011.

[10] H. Lindgren and P. Winnberg. Evaluation of a semantic web application for collaborative knowledge building in the dementia domain. In *eHealth*, pages 62–69, 2010.

[11] P. Lucas. Symbolic diagnosis and its formalisation. *The Knowledge Engineering Review*, 12:109–146, 1997.

[12] J. C. Nieves, M. Osorio, and U. Cortés. Semantics for Possibilsitic Disjuntive Programs. *Theory and Practice of Logic Programming*, Available on doi: 10.1017/S1471068411000408, 2011.

[13] J. O'Brien, D. Ames, and A. Burns, editors. *Dementia*. Arnold, 2000.

[14] V. L. Patel, D. R. Kaufman, and J. F. Arocha. Emerging paradigms of cognition in medical decision-making. *Journal of Biomedical Informatics*, 35(1):52–75, 2002.

[15] H. Prade. Advances in data management. In *Current Research Trends in Possibilistic Logic: Multiple Agent Reasoning, Preference Representation, and Uncertain Databases*. Springer Berlin / Heidelberg, 2009.

[16] D. Silverman. *Interpreting Qualitative Data*. SAGE Publications, 2006.

Towards an implementation of a social electronic reminder for pills

Ignasi Gómez-Sebastià, Dario Garcia-Gasulla, Sergio Alvarez-Napagao,
Javier Vázquez-Salceda and Ulises Cortés

Departament de Llenguatges i Sistemes Informàtics. Universitat Politècnica de Catalunya (BarcelonaTech-UPC)*
igomez,dariog,salvarez,jvazquez,ia@lsi.upc.edu

ABSTRACT

Non-compliance with prescribed medications is a major problem for elder people living alone in developed countries. Forgetfulness and confusion can lead to it, specially when multiple pathologies require a cocktail of different medications each delivered at different time intervals during different periods of time. Assistive technologies, a recent application area for a wide range of Artificial Intelligence techniques and tools, have been effectively used for supporting people in their daily activities. This paper introduces the design and implementation of a system for assisting elder people on following the treatment prescribed by a professional, with the novelty of being based in social and organisational aware assistive technology.

Keywords

Assistive Technologies, normative agents, agent-oriented software design, Ambient Intelligence

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Human Factors—*Assistive Technologies, normative agents, agent-oriented software design, Ambient Intelligence*

1. INTRODUCTION

In 2002 it was estimated [19] that the European population above 65 years old represented between a 12% and a 17% of the total population. Over the next few years, this proportion is likely to increase due to a decreasing birthrate. At the same time, the cost of supporting an elder person is greater than the cost of supporting a child in a ratio of five to three [20], most of this cost being caused by higher health expenses. In the coming years this situation (together with other economic factors) will put great pressure on the national healthcare budgets, mainly because therapies for managing chronic diseases (*e.g.*, diabetes, parkinson, *etc.*) are performed away from the institutional care setting (typically at home). This distributed approach to daily care requires that elders are capable of autonomously taking several different medications at different time intervals over extended periods of time. This can easily lead to forgetfulness or confusion when following the prescribed treatment, specially when the patient is suffering multiple pathologies that require a treatment with a cocktail of drugs. This gets worsened when elders suffer a cognitive impairment.

Assistive Technologies (AT) are an application area where several Artificial Intelligence techniques and tools have been successfully applied to support elder or impeded people on their daily activities. However, approaches to AT tend to center in the user-tool interaction, neglecting the user's connection with its social environment (such as caretakers, relatives and health professionals) and the possibility to monitor undesired behaviour providing both adaptation to a dynamic environment and early response to potentially dangerous situations.

AT can be effectively used for guiding elders with their prescribed treatments, avoiding major problems such as noncompliance with the treatment and adverse drug reaction. Several devices are available for helping patients to manage their daily doses of medication. They range from simple pill containers with multiple compartments that can hold a month's supply to intelligent pill dispensers [13] with an alarm function which can detect when the patient takes the pill, and that can be telematically programmed in case the treatment changes. However, these kinds of devices tend to have a static encoding of their functions, and are unable to react to changes in the environment (*e.g.*, they will keep on dispensing the pills even if the patient is on holidays away from home) and autonomously react to potentially dangerous situations (*e.g.*, the dispenser is about to run out of supply for a given pill). Furthermore, to the best of our knowledge none of these devices take into consideration the important role that third parties may have in the activity. For instance, the prescribing doctor scheduling a visit with the patient when the treatment finishes, a delivery company refilling the dispenser when it is about to run out of medication, or patient's personal computer displaying reminders when it is time to take a given medication. Nor they reflect the social constraints that apply in the relation between the user and the other actors. For instance, forbidding the delivery company employee from entering user's home if the doctor considers the user can fill the dispenser autonomously.

The COAALAS [10] project (COmpanion for Ambient Assisted Living on Alive-Share-*it* platforms) is based on organizational and normative theories and Ambient Assisted Living (*i.e.*, ambient intelligence applied to assistive technolo-

*C/Jordi Girona 1-3, E-08034, Barcelona, Spain

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), June, 4–8, 2012, Valencia, Spain.

gies). The project aims to create a society of organisational aware devices (typically sensors and actuators) that are able to adapt to a wide range of Ambient Assisted Living situations. COAALAS models the device network around the user as a society, including the set of behavioural patterns the devices are expected to follow. COAALAS effectively supports smart assistive tools that integrate human actors with the surrounding devices, contributing to the state-of-the-art in semi-autonomous and intelligent devices for elder people by allowing the devices to be both social and norm aware.

In this paper we present the design and proposed implementation of a social-norm aware pill dispenser. The dispenser, based on the concepts developed by the COAALAS project, will help the elderly or disabled people to manage their daily doses of medication while presenting the following three properties:

- *Social awareness*: The device is connected with other assistive devices and with relevant actors (such as doctors, caretakers and other health professionals, familiars, etc) for helping the elder take his daily doses of medication.
- *Autonomy*: The device can react to changes in the physical or social environment without requiring human intervention. Furthermore, it should be able to react to simple changes in the scenario autonomously (*e.g.*, a change in the scenario implies the pill dispenser is not filled by the patient any more, but by a care giver).
- *Normative awareness*: The device performs its task while following a set of specified behavioural patterns. However, due to its autonomy, the device has the option of breaking the patterns, provided it considers it will be in the benefit of the society (*e.g.*, if an incoming stock break is detected).

The rest of this paper is structured as follows. First a short survey on existing works for assisted living, focused on these works that facilitate taking a prescribed medication, is presented. Then, one of such existing works, the COAALAS project is explained in depth. Later, a use case is introduced and modelled using the ALIVE methodology which one of the theoretical foundations of the COAALAS project. Finally, conclusions are drawn.

2. STATE OF THE ART

This section presents a short survey on the existing work in the area of Ambient Intelligence for supporting independent living, with special emphasis on the works focused on facilitating daily tasks (specially taking a prescribed medication). Special attention is put on the COAALAS project, that has been selected as basis for the work presented in this paper.

Robocare [6] is a project deployed on a domestic test-bed environment that combines a tracking component for people and robots and a task execution-supervision-monitoring component. Robocare has the goal of contributing to improving the quality of life of elder people living autonomously in their homes. The system is composed of several software and hardware agents, each providing a set of services, and an event manager that processes requests to the different services and directs them to the appropriate agents. The system also includes a monitoring agent, with knowledge of the assisted person's usual schedule. The monitoring agent is able to detect discrepancies between the tasks performed by the user and the expected schedule and react to them. In order to coordinate all the agents and monitor user's behaviour heavy computational processes take place, limiting the tested scenarios to non-crowded environments, where only 2-3 persons and only a small portion of the domestic environment are monitored. What is more, the expected schedule is non dynamic and small justified deviations (*e.g.*, relatives visiting the user) are currently detected and corrected.

The AHRI (Aware Home Research Initiative) [16] is a residential laboratory for interdisciplinary research where several projects have been evaluated. The most relevant one is the ISLA (Independent LifeStyle Assistant) [12] project, that passively monitors the behaviours of the inhabitants of the residential laboratory, alerting relatives in case of potentially dangerous situations (*e.g.*, the user falls). The ISLA project presents two main innovations with regards to the Robocare project:

- Agents autonomosly interact within them in order to achieve their goals, without the need of an event manager agent that coordinates them. However, in order to transform context-free perceptions provided by the agents into context-aware perceptions, a centralized coordinating agent is used.
- Agents are able to learn schedules based on the daily tasks performed by the inhabitants. Models are built, reflecting which devices are triggered when given activities are performed, and alerts are raised whenever an unlikely activity takes place. Therefore, instead of using generic static schedules for the users, the schedules are built dynamically based on user's detected behaviour. However, once a schedule has been learned, user is not able to deviate from it without raising an alarm.

Evaluation of the ISLA project presents two main conclusions:

- The need for coordination of the agents and centralized control outweights the benefits of the distribution and independence of components agents architectures provide.
- Partial observability of actions performed by the inhabitants is a problem, specially when plans are abandoned due to forgetfulness and reminders need to be issued. Inhabitants do not tend to be in favour of having every of their moves observed.

In the scope of the MINAmI project [15] a qualitative study of three ambient intelligence scenarios is reported, being the most relevant one a scenario that deals with monitoring the taking of medication. In the scenario users are given a smart pillbox, with a cap that counts the number of opening and closing events and a clock. The pillbox can communicate with a mobile phone, that displays the timed record of cap openings and closings. If the users forgets to take his medication for a prolonged period of time, the pillbox sends a notification to a care center. During the evaluation of the scenario, users felt it was too intrusive on their

privacy, arguing the data should not be reported to their doctors. They considered relying on such devices for the reminders could weaken people's cognitive abilities, and that such a system would not be suitable for users taking a cocktail of medication rather than just a single medicament, as several pillboxes should be provided. The scenario presented seems to be mainly theoretical, lacking an implementation, and does not provide a fully integration of the pillbox with the rest of the devices in the Smart Home (*e.g.*, the system can notify that the user forgot to take his medication even when the rest of the devices are showing that the user has not been at home on the last 3 weeks, for instance, because he is on holidays).

In [5] ECA (Event-conditioning-action) rules are used for Smart Homes that support assisted living for the elderly. A basic interpretation of the ECA rules is that, on detecting certain events, if certain pre-conditions are satisfied, then a given set of actions are to be enacted. By using rule-based systems and other AI techniques, devices and hardware-oriented technologies for Smart Homes can be augmented and enriched. With that goal in mind, authors propose connecting the devices to a central monitoring facility that performs all the reasoning. This approach differs from the resnt in the sense that devices show a complete lack of intelligence, leaving all the reasoning to a central component, effectively preventing coordination and cooperation among the agents representing the different devices. A similar work [18] proposes using abductive logic programs for the reasoning process. Abductive logic programs provide active behaviour, just like the ECA rules, but they also provide added declarative semantics and a extensive background knowledge available via the logic programming. For instance, this approach allows for easily applying preferences to the reminders issued to the user. Both works present a higher system adaptability, allowing even for a customization that adapts the system to the preferences of the user. However they lack the coordination among different agents that would allow the system to autonomously recover from a failure if one of the agents stops working.

In COAALAS (COmpanion for Ambient Assisted Living on Alive-Share-*it* platforms), organizational and normative structures are used to model the device network around disabled users as societies, along with the expected behavioural patterns, effectively supporting smart assistive tools that integrate with the human actors around them. The project aims to make the devices intelligent enough so they can autononosmly organize, reorganize and interact with other actors. This intelligence allows the integration of the devices in a Smart Home with the rest of actors involved with the user (*e.g.*, doctors and other health professionals, caretakers, *etc.*), and to embed the devices with autonomous, proactive, social and adaptable behaviour. The COAALAS project puts the theoretical basis for modelling and deploying assistive scenarios as societies of cooperating adaptable norm-aware actors. This approach makes COAALAS suitable for two main scenarios:

- Dynamic scenarios. Having adaptable actors implies the system design does not require major adaptations when changes in the scenario occur. For instance, if a new actor is to be added or the actions an actor performs are to be enacted by a different actor, changes on the system design are few and simple. Even more, changes on the system implementation can usually be completely avoided. Therefore, in dynamic scenarios where changes to the design are to be applied often, COAALAS allows for a fast and swift adaptation to the new scenario.
- Scenarios where the actors (specially artificial ones) have to be norm aware. Actors in COAALAS follow a set of expected behavioural patterns by default, but with the possibility to temporally stop following them if they consider it is beneficial for the society or the individual. Therefore, COAALAS facilitates the specification and implementation of normative constraints, provided they are important for the scenario being developed.

However, the COAALAS project only presents a theoretical basis and no implementation exists. We choose COAALAS to base our work as we intend to provide a proof-of-concept implementation of a dynamic scenario with flexible behavioural patterns that can be easily modelled as a society of cooperating actors.

3. THE COAALAS PROJECT

The main goal of COAALAS is to contribute to the state-of-the-art in semi-autonomous and intelligent devices for elder people. COAALAS builds on the results of two European funded projects: EU-Share-*it* [4] and EU-ALIVE [1]. COAALAS aims to produce a new generation of Ambient Intelligence devices for elder people by embedding several state-of-the-art AI techniques:

- **Autonomy**: The device is integrated in the environment, able to perceive it and react to it in a timely fashion.
- **Proactivity**: The device is able to anticipate and take the initiative in order to fulfil its design objectives.
- **Social behaviour**: The device is integrated in a community of actors and is aware of social regulations and protocols.
- **Adaptability**: The device will modify its behaviour based on the rest of the actors around it.

By a combination of these techniques, COAALAS focuses in making devices intelligent enough to organize, reorganize and interact with other actors. Devices have an awareness of their social role in the system – their commitments and responsibilities – and are capable of taking over other roles if there are unexpected events or failures. The objective of the COAALAS project is to create a society of physically organisational-aware devices able to adapt to a wide range of Ambient Assisted Living situations that could have an impact on the user's well-being.

3.1 The ALIVE architecture

The ALIVE framework presents a multi-level structure that combines model-driven design techniques and agent-based system engineering providing support for *live*, open, and flexible service-oriented systems. ALIVE's organisational and normative structures make it suitable for highly regulated scenarios like the one presented in this paper. The ALIVE framework applies substantive norms that define commitments agreed upon actors and are expected to be enforced by authoritative agents, imposing repair actions and

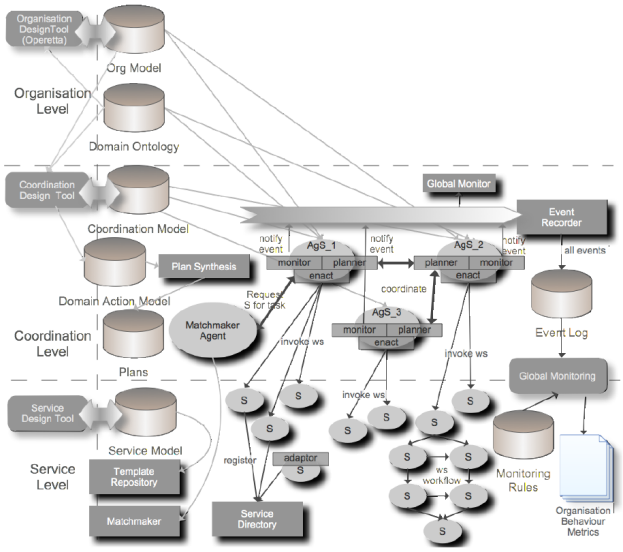

Figure 1: ALIVE architecture [2] (S stands for Service)

Property	Value
Activation Condition	isActive(Provide_medication)
Deadline	
Expiration Condition	isFullfilled(check_best_before_date)
Maintenance Condition	~isFullfilled(Provide_medication)
Norm ID	o_check_best_before_date_before_Provide_medication

Property	Value
Activation Condition	isViolated(O_check_best_before_date_before_fill_dispenser)
Deadline	
Expiration Condition	isFullfilled(Apply_generic_sanction)
Maintenance Condition	
Norm ID	o_Apply_Generic_Sanction_fill_dispenser

Figure 2: Example of norms of the use case

sanctions if invalid states are reached. Substantive norms allow the system to be flexible, by giving actors (human or computer-controlled) the choice to cause a violation if this decision is beneficial from an individual or collective perspective.

The ALIVE organizational level represents the organizational structure of the system via the organizational model. The organizational model is formalized following the Opera methodology [8], including the following concepts:

1. **Objectives**: States of the world pursued by actors. Derived from organisation's goals.

2. **Roles**: Groups of activity types played by actors (either agents or human users). The set of roles and the relations between them constitutes the *Social Structure*.

3. **Landmarks**: Represent important states of the world regarding the achievement of goals. They are identified by the set of propositions that are true on the state of the world represented by the landmark.

The organisation level supports the definition of norms, adding a normative structure to the social and interaction structures. The normative structure is useful for imposing patterns of behaviour. The elements on the normative structure contain the following main components, expressed using Partial State Descriptions of the world: a) *Activation Condition*: when the world reaches the state specified in this condition, the norm starts to be checked. b) *Expiration Condition*: when the world reaches the state specified in this condition, the norm stops to be checked, and has not been violated. c) *Maintenance Condition*: when the world reaches the negation of the state specified in this condition, the norm stops to be checked, and has been violated. *Figure* 2 shows an example of an ALIVE norm.

The ALIVE coordination level provides actors' patterns of interaction. The organisational model is transformed into a repository of coordination plans. Plans bring the system from the state represented by a landmark to the next one (as defined on the interaction structure) and are formed by chains of tasks. The model of tasks contains both pre and post-conditions, that define the state of the world before and after a particular task is performed, and the permissions (roles in the organisational model) required for executing the task. Being organizational aware, agents can select one or several roles according to their capabilities (*e.g.*, the tasks the agent can perform) and start enacting the plans associated to that role as required.

The ALIVE service level maps actions in the environment to abstract tasks. Non-organizational aware agents in the system register their capabilities (*e.g.*, tasks they can perform) via a white pages system and are coordinated by the organizational aware agents to execute the tasks required for enacting the different plans.

The monitor tool is the back-bone of the ALIVE framework, connecting all the three levels allowing the exchange of events among them, from a action on the environment that fails to an update on the Organisational design (*e.g.*, a new role or objective is introduced) that affects the agents in the coordination level. Agents enact their roles by interacting among them via direct communication (coordinating among themselves) or by interacting with the environment. The monitor tool observes these interactions and matches them with the normative and organisational states (*e.g.*, Obligations, Permissions, Roles) effectively allowing agents to reason about the effects (in a normative sense) of their actions.

4. EXTENDED AND GROUNDED USE CASE

The COAALAS project has a main milestone: solving the difficulties presented in the use case described in [11]. Such use case was originally presented in [3]. For coherency with the COAALAS project, our design will also be based on that use case. This will allow us to state how far can we take the implementation of the COAALAS project.

The original scenario focuses on an elder or disabled person with difficulties to leave his house. It is assumed that such person is following one or more medical treatments which require periodical doses of medication. The main problem to solve is that of supplying the required stock of medicines to the subject while supervising that he follows the medical treatment prescribed by his doctor, not missing any dose due to forgetfulness or taking it at the wrong time due to confusion. It must be remarked that the design presented in this paper includes some minor modifications with respect to the original COAALAS scenario in order to make the proposed implementation feasible. *Figure* 3 shows the interfaces of the different actors. These are the main actors

in our scenario:

- *User*, as the main character of the scenario. Responds to medication notifications sent by the *Medical dispenser* actor. Provides access to user's calendar in case other actors (typically *Doctor* and *Caretaker*) need to query it.

- *Doctor*. Responsible for prescribing the medical treatment. When the treatment is about to expire, the *Doctor* is notified and must decide to: 1) continue with the same treatment. 2) continue with a modified treatment (*e.g.*, different doses or medications). 3) schedule a visit with the *User* to control his evolution and assess his state. The *Doctor* has the commitment of deciding in a reasonable period of time.

- *Caretaker*, as the responsible for looking after the patient. If a potentially dangerous situation is detected, the *Caretaker* is notified. Then, the *Caretaker* has the commitment to go to the user's place to check the anomaly. This process implies acknowledge of the notification and a textual report of the solution provided for the anomaly. The *Caretaker* can also perform tasks in substitution of another actors that are temporally unable to perform them. For instance, the *Caretaker* can take medication to the user's home in case the logistics company is unable to deliver it on time for the treatment.

- *Health Insurance Company*, as the entity organizing and controlling the interactions. The *Health Insurance Company* applies sanctions in case commitments are not fulfilled. It also coordinates other actors in order to enact repair actions (*i.e.*, actions that take the state of the world from a potentially dangerous situation to a safe one). For instance, the *Health Insurance Company* can ask a *Caretaker* to bring medication to the user's place if the *Logistics company* is delayed and will not be able to deliver the medication on time for the treatment. Finally, it coordinates other actors in the most efficient way possible, planning and optimizing medication delivery routes.

- *Pharmaceutic*, as medication retailer. Pharmaceutics provide medication to logistic company employees so they can take it to the user's place. They have the commitment to avoid providing medication until the logistic company employee has successfully authenticated. *Pharmaceutics* receive notifications on user's medication stock and consumption, so they can plan ahead of medication consumption and coordinate among them in order to satisfy medication supply needs. Pharmaceutics provide reports to the *Health Insurance Company* with relevant information on their current stock of medications. The *Health Insurance Company* actor uses this information for choosing the pharmacies that will provide the medication based on several efficiency terms.

- *Logistics company*, as the responsible for home delivery. Logistics company employees authenticate at pharmacies in order to take medication from pharmacies to user's place. If they are granted access to user's place (because the *Doctor* considers the *User* is not able to refill the *Medical dispenser* on his own) they have the commitment to refill user's *Medical dispenser*. Otherwise they have the commitment to avoid entering user's place. They can issue delay notifications in case they realize they will not be reaching user's place at the expected time.

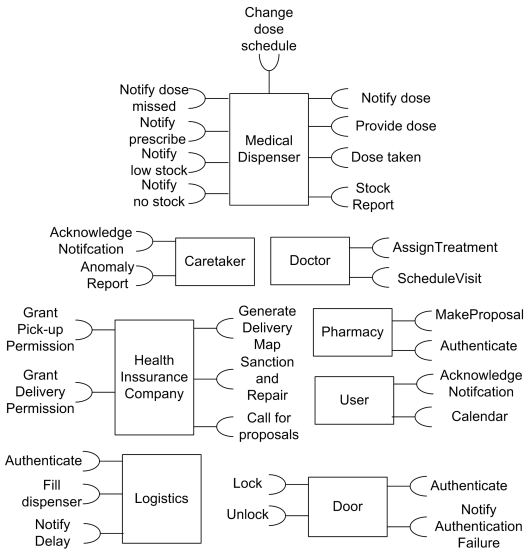

Figure 3: Interfaces of the different actors

- *Domotic house*. In the current scenario only domotic doors are taken into account. We plan adding more domotic actors in the future. The *domotic door* is the actor responsible for allowing other actors access to the user's place (typically *Caretaker* and *Logistics company*). Actors authenticate at the *domotic door* and the door will allow access provided actor can get into user's home, denying it otherwise, and contacting a *Caretaker* in case too many unsuccessful authentication attempts are performed.

- *Medical dispenser*, a device which provides medication doses. The *Medical dispenser* is able to provide pills at scheduled times and asses, with an *acceptable* degree of certainty, weather the user has taken the pill from the dispenser or not. An intelligent controller attached to the dispenser can coordinate with another actors (typically *Doctor* and *Caretaker*) to change the schedule of medication doses, or do it autonomously (*e.g.*, providing more pills if an interpretation of user's calendar indicates he will be away from home when the medication should be provided, so the user can take the pills with him at take them at the scheduled time). The controller can send reminders to user's smart-phone when it is time to take the medication, control the stock of medicines (autonomously coordinating with the *Health Insurance Company* to schedule more medication deliveries) and react to unexpected events (*e.g.*, user not responding to reminders).

The original COAALAS scenario contemplates a wide set of situations, but the main one can be outlined as follows:

The *User* visits the *Doctor* who assigns him a medical treatment. The *Health Insurance Company* receives a notification about the treatment starting, and contacts the different *Pharmaceutic* actors near user's place. The different *Pharmaceutic* actors reply with offers that state the amount of medication they can provide and when. Before replying, pharmacies can coordinate among them in order to provide group offers that are usually better than the offers they could provide individually. The health insurance company chooses the best offer based on several efficiency parameters (such as amount of medication provided, delivery time, expiration date, price, *etc.*) and sends a notification to the pharmacy or set of pharmacies selected. Then, generates a visual map with the location of the pharmacies and the user's place and provides it to the *Logistics company* actor. Finally, the *Health Insurance Company* grants the logistics company authorization to retrieve the medication from the pharmaceutic and to enter the user's house, provided it is required. The logistics company authenticates at the selected pharmacies picking the medication and goes to user's home in order deliver it, or put it on the medical dispenser unit. If the medical dispenser is to be refilled, the logistics company employee authenticates via the *domotic door* actor. An alert is sent to the *Caretaker* actor if too many failed authentication attempts are detected. Once filled with medication, the medical dispenser provides medication doses and notifies the *User* when a dose is to be taken. The *Medical dispenser* actor controls the stock of medications. For doing so, it takes into account the prescribed treatment (*i.e.*, both length and daily doses) and the number of available doses. If the stock of medications is low, the medical dispenser contacts the *Health Inssurance Company* actor, so a new delivery can be scheduled. If the treatment is about to prescribe, the medical dispenser contacts the *Doctor* actor. Then, the doctor can choose to continue with the treatment, modify it or schedule a visit to control user's evolution. Both the *Medical dispenser* and the *domotic door* actors can contact the *Caretaker* if a potentially dangerous situation is detected. Such situations include, someone entering the user's place without authorisation, running out of medication stock and user not responding to dose taking notifications. In order to check some potentially dangerous situations the *Caretaker* must enter the user's place. Therefore, both the *domotic door* and the *Medical dispenser* actors can grant the *Caretaker* actor access to the user's place.

From this simplified use case we extend the interactions and define a set of technologies to implement them. In our design, the authentication of the actors at the pharmacy and the domotic house is performed via RFID. Both caretaker and logistic company actors carry RFID tags with them. The tags can be used to authenticate the user at the readers on the pharmacy, domotic door and pill dispenser.

The communications required to coordinate the different actors in the use case are implemented via social networks, particularly Tweeter. The already available APIs[1] for Tweeter will facilitate their integration and the development of the proof-of-concept prototype. Even though no personal relevant information is exchanged, messages can be encrypted using a public-private key pair paradigm if privacy issues have to be taken into account. *Figure* 4 depicts the different messages exchanged in the scenario. These are the

[1]`http://twitter4j.org/en/index.html`

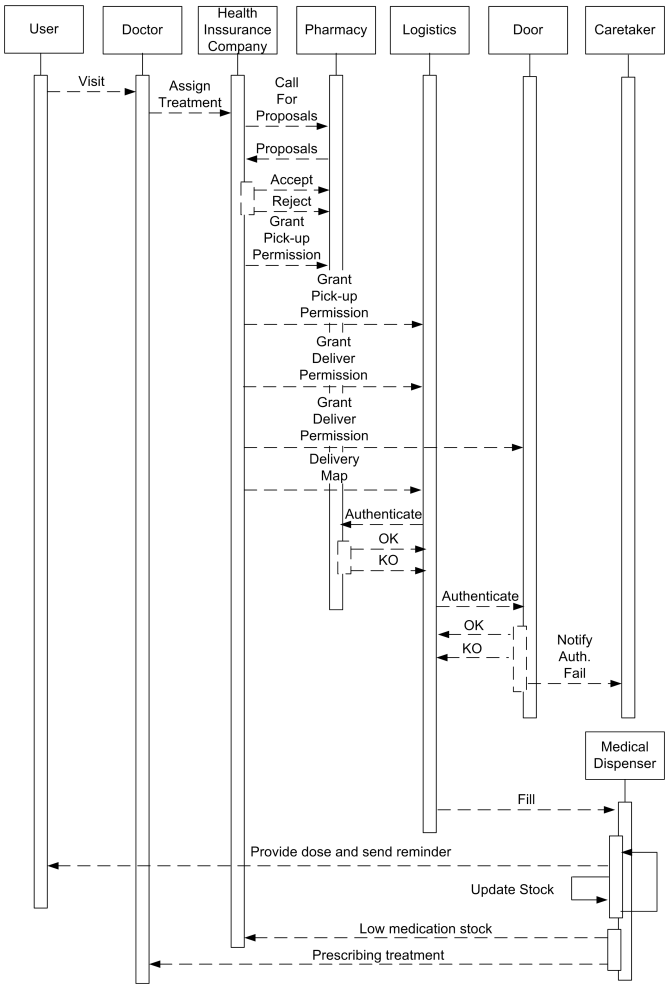

Figure 4: Sequential diagram of message exchange

main communications to be implemented via social network:

- When the *Doctor* receives the visit of the *User* and prescribes a treatment, he sends an encrypted message to the *Health Insurance Company* with information on the prescribed treatment and the medication required to follow it.
- Aware of the treatment user has to follow, the *Health Insurance Company* contacts different pharmacies, asking if they have the medications in stock. Pharmacies coordinate among them, returning different individual or collective proposals to the *Health Insurance Company*. Once the best proposals are selected, acceptance or refusal notifications are sent to the pharmacies.
- Pick-up permission notifications are provided to the selected pharmacy (or set of pharmacies) and the logistic company. if the *Doctor* consider the *User* needs help refilling his medical dispenser unit, delivery permission notifications are sent as well. Finaly, a message including the delivery route to be followed is provided.
- The *logistics company* actor authenticates at the pharmacy and at the domotic door by providing authentica-

tion messages that include his RFID tag. It allows the actor to pick-up the medication and access user's place respectively. If too many authentication attempts are performed at the door, *Caretaker* receives a notification message.

- The *logistics company* actor fills the dispenser and notifies it with a message to the *Medical Dispenser* including relevant information on the medication provided (such as quantity, format, expiration date, *etc.*).
- The *Medical Dispenser* starts providing doses and sending reminders to the user according to a schedule. Stock is updated accordingly. Please notice stock update is not a message, but an action performed internally by the *Medical Dispenser* actor.
- When the *Medical Dispenser* is almost out of stock for an on-going treatment, it will send a message to the *Health Insurance Company* so that it can organize and schedule a delivery. If the situation is critical (for instance, being completely out of stock) the *Medical Dispenser* can send a message to the *Caretaker* instead, so he can get some medication doses quickly to user's place.
- When the treatment is about to prescribe, the *Medical Dispenser* sends a message to the *Doctor*, and the doctor decides to continue with the same treatment, modify the current treatment or schedule a visit with the user to control his evolution.
- If the *Medical Dispenser* detects that the daily dose has not been taken (typically, user does not respond to notifications), it will send a message to the *Caretaker* to let them now of the potentially dangerous situation.

In our design we also include several Web applications of integrated systems, which will provide additional features to the system. An example of that is Google Maps in the following case. When a new delivery is to be arranged, the Health Insurance Company will choose the pharmacy which is to provide the medications. That selection will be made by a Call For Proposals (CFP) to determine the best candidate considering a set of parameters (*e.g.*, location, medication stock, best before date of the medication required considering the treatment). Once the set of pharmacies are decided, the Health Insurance Company will provide the logistics company with a Google Maps of both the pharmacy and the house where the delivery is to take place. Another example of integrated web applications is connecting the medical dispenser to the user's on-line calendar (*i.e.*, via xcal). If the dispenser detects the user will be away from home (*e.g.*, the user is on holidays) it will stop dispensing pills and reminders will be forwarded to user's cell-phone (*e.g.*, via SMS).

Regarding the autonomous agents and its devices, we have decided to implement them using the newly marketed tiny computers. Concretly we will use *Raspberry Pi*[2], a credit-card sized computer with Ethernet, USB and HDMI ports, and Micro USB for power. The agents to be hosted by these computers are the domotic house (for the RFID reader) and the medical dispenser (for the embedded intelligence). That will allow them to communicate with each other and the rest of the agents through a network. Once the hardware is integrated into the organization, more agents and devices can be easily added (*e.g.*, motion sensor, temperature sensor) to further enhance the monitorization of the patient. Currently we are implementing a REST-based interface to connect lightweight agents with a set of sensors with Arduino and Phidgets interfaces. The preliminary diagram of the domotic house is depicted in Figure 5.

For the user interface with the system, we have decided to use a smart-phone with an Android based operative system, as well as a tablet PC. The former will allow the user to receive notifications and control his calendar, so it can be integrated with the caretaker and the medical dispenser actors. In case the user is away from home, the smart-phone can be used to track user's position, in case the doctor considers it is required. The latter will be used as a guidance interface, both by means of electronic tutorials [17], and as an audiovisual communication interface with doctors and nurses.

Finally, regarding the medical dispenser two out-of-the box solutions are analyzed, the Simple Med [14] pillbox from the *Vaica Medical* corporation in Tel Aviv and the *u*Box [13] from IIH (Innovators in Health). The Simple Med pillbox is a unit with a grid of boxes (7 days a week 4 doses a day) that can be plugged into the phone line and programmed by a provider to call the patient and notify which compartment to take the pills from. The Simple Med pillbox is able remind the patient (visually and audibly) to take his medication at the right moment of time. The system can be connected to an external monitoring center, sending signals in response to box opening and closing events as well as alert signals in response to recognized deviations (typically missed doses and doses taken at the wrong time). The *u*Box is a palm-sized pill dispenser that reminds a patient when it is time to take medication and records when the patient takes a pill from the dispenser. It is also able to track accesses to the box by program personnel (typically a health professional refilling the box) via RFID keys. Currently, the *u*Box is being tested in three districts in Bihar, India [9]. Comparing both solutions, the *u*Box seems a better approach for the implementation of our scenario as it provides a better connectivity (allowing for easier integration with the tiny computer) at the cost of lower integrated intelligence that we are not going to use anyway, as our custom controller module is going to be attached to the pill dispenser.

4.1 Modelling the use case using the COAALAS architecture

This section models the use case deployed in this paper using the COAALAS architecture that is based on the ALIVE framework. It introduces the model from the ALIVE Organisation, Coordination and Service level perspectives. Elements from the domain ontology are referenced when required.

4.1.1 Organisation level model

In the organisation level we identify the existing set of stakeholders, the goals of each of them, landmarks and scenes related to those goals and the normative structure of the system. The stakeholders map directly to the different actors in the use case. A role is created for each actor.

For each *role*, its *goals* within the use case have been iden-

[2]http://www.raspberrypi.org/

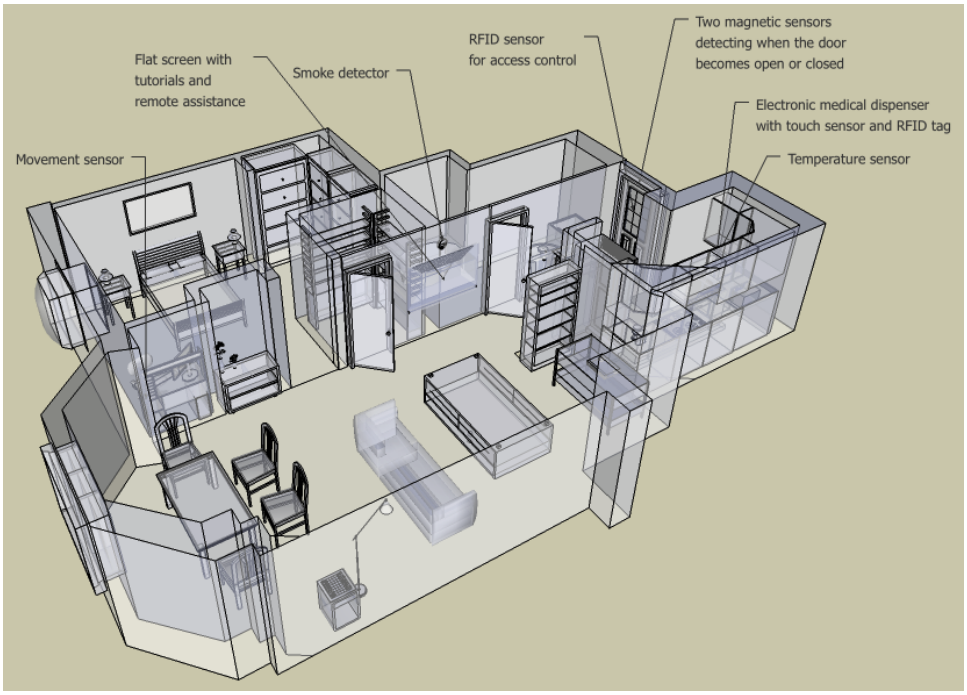

Figure 5: Plan of sensorisation for the COAALAS domotic house

tified. Also a hierarchical relation representing dependencies between roles is defined. For instance, to fullfil the goal *Prepare_deliver*, pick-up and deliver permissions must be assigned to both *Logistic* and *Pharmacy* roles. Thus, the role fulfilling *Prepare_deliver* depends on the role assigning permissions. Each goal is assigned with a state description, representing the state of the world when the objective has been fulfilled. For example, the goal *Reach_pharmacy* has been fulfilled when the logistics person A (who has been given the order to go to a given pharmacy P) is in the same geographical coordinates as P.

Landmarks define important states on the achievement of the goals. A set of ordered landmarks define a scene. The execution of each scene entails the achievement of one or more goals. For instance, when the landmark *Check_treatment* is reached, the goal *Aprove_treatment* is fulfilled. Role *Doctor* participates on the landmark because it has the objective *Aprove_treatment* assigned.

The last element to be defined on the organisation level are the norms, which must be specified by an expert on the domain together with the supervising entity, if there is one (in this case it would be the Health Insurance Company).

4.1.2 Coordination level model

The elements on organisation level are derived into the coordination level. This level contains three main elements, actions, plans and organisational aware agents. Organisational aware agents enact roles based on their capabilities (*i.e.*, actions they can perform) and the goals associated to that role. Actions are derived from objectives on the organisation level. Actions contain both pre and post-conditions representing the state of the world before and after the task is enacted. Thus, the post-condition of a given task matches the state description of the objective the task is derived from.

Plans represent chains of actions. Typically, one plan is modelled for each transition between landmarks or scenes defined on the organisation model. In the case of the transition between the scenes $Prepare\_request\_medication \rightarrow Request\_medication$ and $Pick\_up\_medication \rightarrow Hand\_medication\_over$ one additional plan is provided for each transition. In this case, the agents have two options available in order to bring the world from the state represented by the first scene to the state represented by the second one. The norms defined on the organisation model provide means for pondering both plans (analyzing which plan will cause less norm violations) and choosing the most appropriate one on each situation.

4.1.3 Service level model

This model consists in a set of OWL-S annotations applied to the web services available in the system. These annotations, modeled in the form of Inputs-Outputs-Preconditions-Effects allow the matchmaker component to find suitable services for the tasks the agents try to enact. This intermediate component allow a dynamic mapping between the tasks and the services, choosing the most appropriate service on the fly. This means different services (with equivalent descriptions) can be chosen for the same task depending on service's availability and performance. For instance, the task *identify_delivery_person* can be performed by enacting the services *id_card_reader* or *domotic_door_iris_reader*.

5. CONCLUSIONS

In this paper we have presented our agent-based approach for assisting an elder in daily routine tasks related to his/her medication needs. The system presents a flexible multi-level architecture able to model the complex interactions among different actors involved in the assisted tasks (see *Section* 4). Actors have different responsibilities and offer or consume

different services. Due to the connection among levels, a change in the organisational level can trigger changes both in the coordination level (*via* plan and agent generators) and in the service level (new plans will result in the execution of new tasks and, possibly, the invocation of new services).

Thanks to this approach, new roles, objectives and norms can be introduced in the organisational level, without designer having to perform any modification to the coordination and service levels, as these changes are automatically performed. The system introduced allows for monitoring the different actions performed by the set of actors in order to fulfill the assisted tasks. Deviations from the original plan can be detected, and sanctions or repair actions applied (*e.g.* sending a doctor, or an urgent shipment of medications to the patient).

Intelligent agents, at the coordination level, are an option for providing both exception handling and organisational-normative awareness capabilities to the system. Exception handling is common in other service-oriented architectures, however, most approaches tend to focus on low-level (*i.e.* service) exception handling. The ALIVE approach enables managing of exceptions at multiple levels either substituting services (service level) looking for alternative workflows to connect two landmarks (coordination level) or even looking to achieve alternative landmarks among the same scene (organisational level). Agents at coordination level enable this medium and high-level exception handling, which are not commonly seen in other service-oriented approaches. Regarding organisational normative awareness, making normative agents reason about the workflows (and the tasks included in them) before performing them, and discarding the ones that do not comply with organisational norms, adds organisational awareness to the execution of the workflows. Normative agents come in handy on the presented case, as they can perform reasoning about what actions to perform taking into account both the actions available and the norms defined.

Assistive technologies are applied to support people in their daily life. Most approaches focus solely on the direct interaction between users (in our case, disabled patients) and the assistive tool, but AI has the potential to provide innovative mechanisms and methods capable of taking into account more complex interactions. For instance, such an approach can take into account the important role that third parties may have in user activities, and explicitly reflect the social constraints that apply in the relationship between device and patient. In COAALAS (COmpanion for Ambient Assisted Living on Alive-Share-*it* platforms), organizational and normative structures are used to model the device network around disabled users as societies, along with the expected behavioural patterns, effectively supporting smart assistive tools that integrate in perfect harmony with the humans around them. The result is an assistive society of ambient-aware assistive tools.

The main goal of COAALAS is to contribute to the state-of-the-art in semi-autonomous and intelligent devices for elderly people. The target population for these supporting devices includes individuals who are independent enough to live autonomously in their community. The role of intelligent devices is to maximize their safety and comfort, thus increasing their quality of life and delaying their institutionalization.

Using a combination of these techniques, COAALAS focuses on making devices intelligent enough to organize, reorganize and interact with other actors. Devices have an awareness of their social role in the system – their commitments and responsibilities – and are capable of taking over other roles if there are unexpected events or failures. In short: our objective is to create a society of physically organizational-aware devices able to adapt to a wide range of Ambient Assisted Living situations that could have an impact on the well-being of the user.

An important component regarding the final functionaloty of our proposal is the development of a user interface integrated on the user's smart-phone or tablet PC. This interface is key in order to allow users to be separated from all the technical components and low-level processes of the system. First of all, the user interface must be simple enough as to allow elder patients with low technological skills to effectively interact with the system. Secondly, the interface must be expressive enough as to provide more functionalities than the one natively provided by a smart-phone. Finally, the functionalities made available to the user should be adapted to the user cognitive capabilities.

To implement the required functionalities, it is not enough to use native features of smart-phones and tablet PCs. In order to achieve the behavioral requirements of the system (*e.g.*, autonomy, proactivity, social behavior, etc.) it is necessary to extend those native features. For instance, a simple reminder system can be implemented using a smart-phone's calendar and alarm systems. However, it would lack the autonomy, social awareness and normative awareness our proposal provides. A simple alarm system is not able to adapt reminders to user's calendar (reminding user to take a medication dose with him if his calendar indicates he will be away from home when it is time to take his medication) nor is able to alert caretakers if potentially dangerous deviations from user's routine are detected.

We plan to evaluate our approach via tests with disabled patients on a domotic house, as depicted in Figure 5. First, we already had experiences using Agents to control and environment designed for elders at *Casa Agevole* [7]. This will allows us to asses the real potential of intelligent social aware devices, in comparison to other approaches for Assisted Living. Specially when reacting to unexpected potentially dangerous situations. Second, this tests will allow us to evaluate patients' reactions to the usage of the system, assessing the convenience of adding a new actor: a personal assistant agent on the user interface. On the one hand, the personal assistant must be able to show personality up to some extent, in order to improve system's acceptance from the patient perspective. On the other hand, personal assistant must be socially connected to other assistants (*e.g.* for finding patients with similar affections or interests) effectively increasing patient's social integration.

For those cases in which there is special interest in keeping human interactions (*e.g.*, to avoid social separation, because daily care is required), the organizational approach is flexible enough to adapt to those needs. Simply by assigning the role of the logistics company to the caretaker, the latter obtains the required permissions and obligations of the former. In this new scheme, the caretaker would be in charge of visiting the user periodically, filling the medical dispenser when necessary, and providing the user with some social integration. Such solution could be beneficial for certain types of patients without requiring further specification in COAALAS.

6. ACKNOWLEDGMENTS

This work has been supported by the project ICT-FP7-289067 *SUPERHUB*. Ulises Cortés work has also been partially supported by the project TEC2011-29106-C02-02 *Sistema inteligente IWALKER: rehabilitación colaborativa.*

7. REFERENCES

[1] S. Álvarez-Napagao, O. Cliffe, J. A. Padget, and J. Vázquez-Salceda. Norms, organisations and semantic web services: The alive approach. In M. Baldoni, C. Baroglio, J. Bentahar, G. Boella, M. Cossentino, M. Dastani, B. Dunin-Keplicz, G. Fortino, M. P. Gleizes, J. Leite, V. Mascardi, J. A. Padget, J. Pavón, A. Polleres, A. E. Fallah-Seghrouchni, P. Torroni, and R. Verbrugge, editors, *MALLOW*, volume 494 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.

[2] S. Álvarez-Napagao, F. Koch, I. Gómez-Sebastià, and J. Vázquez-Salceda. Making games alive: An organisational approach. In F. Dignum, editor, *AGS*, volume 6525 of *Lecture Notes in Computer Science*, pages 179–191. Springer, 2010.

[3] R. Annicchiarico, F. Campana, A. Federici, C. Barrué, U. Cortés, A. Villar, and C. Caltagirone. Using scenarios to draft the support of intelligent tools for frail elders in the share-it approach. In J. Cabestany, F. S. Hernández, A. Prieto, and J. M. Corchado, editors, *IWANN (1)*, volume 5517 of *Lecture Notes in Computer Science*, pages 635–641. Springer, 2009.

[4] R. Annicchiarico and U. Cortés. To share or not to share share-it: Lessons learnt. In M. Szomszor and P. Kostkova, editors, *eHealth*, volume 69 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 295–302. Springer, 2010.

[5] J. C. Augusto and C. D. Nugent. The use of temporal reasoning and management of complex events in smart homes. In R. L. de Mántaras and L. Saitta, editors, *ECAI*, pages 778–782. IOS Press, 2004.

[6] A. Cesta, A. Oddi, and S. F. Smith. A constraint-based method for project scheduling with time windows. *Journal of Heuristics*, 8:109–136, 2002. 10.1023/A:1013617802515.

[7] U. Cortes, C. Barrue, A. B. Martinez, C. Urdiales, F. Campana, R. Annicchiarico, and C. Caltagirone. Assistive technologies for the new generation of senior citizens: the share-it approach. *Int. J. Comput. Healthc.*, 1(1):35–65, July 2010.

[8] V. Dignum, J. Vázquez-Salceda, and F. Dignum. A model of almost everything: Norms, structure and ontologies in agent organizations. In *AAMAS*, pages 1498–1499. IEEE Computer Society, 2004.

[9] T. P. Foundation. MIT's smart pillbox targets Tuberculosis. `http://www.prajnopaya.org/index.php/component/content/article/39-press-gallery/160-press-article-030308`, 2012. [Online; accessed 02-Feb-2012].

[10] I. Gómez-Sebastià, D. Garcia-Gasulla, and S. Alvarez-Napago. Society of situated agents for adaptable eldercare. *ERCIM news*, (87):23–24, October 2011.

[11] I. Gómez-Sebastià, D. Garcia-Gasulla, C. Barruè, J. Vázquez-Salceda, and U. Cortés. Alive meets share-it : An agent-oriented solution to model organisational and normative requirements in assistive technologies. In *VI Workshop on Agents Applied in Health Care (A2HC)*, Casablanca, Morocco, December 2010.

[12] K. Z. Haigh, L. M. Kiff, J. Myers, V. Guralnik, C. W. Geib, J. Phelps, and T. Wagner. The independent lifestyle assistant (i.l.s.a.): Ai lessons learned. In *In The Sixteenth Innovative Applications of Artificial Intelligence Conference (IAAI-04*, pages 25–29, 2004.

[13] I. I. Health. *u*Box, a Smart Pillbox for TB Treatment. `http://www.innovatorsinhealth.org/solutions/pillbox.shtml`, 2012. [Online; accessed 02-Feb-2012].

[14] V. Medical. The SimpleMed Smart Pillbox. `http://www.vaicamedical.com/en/`, 2012. [Online; accessed 02-Feb-2012].

[15] M. Niemelä, R. G. Fuentetaja, E. Kaasinen, and J. L. Gallardo. Supporting independent living of the elderly with mobile-centric ambient intelligence: User evaluation of three scenarios. In B. Schiele, A. K. Dey, H. Gellersen, B. E. R. de Ruyter, M. Tscheligi, R. Wichert, E. H. L. Aarts, and A. P. Buchmann, editors, *AmI*, volume 4794 of *Lecture Notes in Computer Science*, pages 91–107. Springer, 2007.

[16] G. I. of Technology. Aware Home Research initiative. `http://www.cc.gatech.edu/fce/ahri/projects/index.html.`, 2012. [Online; accessed 02-Feb-2012].

[17] C. Rubio, R. Annicchiarico, C. Barrué, U. Cortés, and C. Caltagirone. itutorials for the support of elderly population with cognitive impairment. In *Proceedings of the 2010 conference on Artificial Intelligence Research and Development: Proceedings of the 13th International Conference of the Catalan Association for Artificial Intelligence*, pages 131–140, Amsterdam, The Netherlands, The Netherlands, 2010. IOS Press.

[18] F. Sadri and K. Stathis. Ambient intelligence. In *Encyclopedia of Artificial Intelligence*, pages 85–91. 2009.

[19] P. D. UN Department of Economic Social Affairs. Population aging. `http://www.un.org/esa/population/publications/ageing/Graph.pdf`, 2002. [Online; accessed 02-Feb-2012].

[20] P. D. UN Department of Economic Social Affairs. World population to 2300. `http://www.un.org/esa/population/publications/longrange2/WorldPop2300final.pdf`, 2004. [Online; accessed 02-Feb-2012].

Position paper: An Agent-oriented Architecture for Building Automation Systems applied to Assistive Technologies

Hannu Järvinen
Department of Media Technology
Aalto University
P.O. Box 15400
00076 Aalto, Finland
hannu.jarvinen@aalto.fi

Dario Garcia-Gasulla
Departament de Llenguatges i Sistemes
Informàtics. Universitat Politècnica de Catalunya
(BarcelonaTech-UPC)
C/Jordi Girona 1-3, E-08034 , Office K2M-201
Barcelona, Spain
dariog@lsi.upc.edu

ABSTRACT

Building Automation Systems typically consists of different types of devices interacting and sharing information with each other. Similarly, Multi-Agent Systems has focused on flexible mechanisms to coordinate networks of (usually distributed) autonomous computational entities. The integration of these two fields provides powerful solutions to complex problems. Such integration has often been attempted by using Web technologies, such as Web services. However, the lack of a common standards among those components has long complicated such integration forcing to the development of translating gateways. WebSocket is a new Web communication standard which solves many of those problems, facilitating the integration of Building Automation Systems, Multi-Agent Systems and Web technologies. To maximize the benefits of each of those three components, we propose a common architecture. We outline our implementation plans in a Health Care use case and exemplify its benefits in that same scenario.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Human Factors—*Assistive Technologies, normative agents, agent-oriented software design, Ambient Intelligence*

General Terms

Design, Standardization

Keywords

Agent-oriented software design, Ambient Intelligence, Assistive Technologies, Building Automation System, Communication protocols, WebSocket, Web Technologies

1. INTRODUCTION

A modern Building Automation System (BAS) should be able to integrate together various devices and subsystems that are communicating using their own specific protocols. As long as this kind of interconnectivity is not achieved at a large scale, further development of the BAS industry is slowed down. Individual attempts have offered solutions for certain sets of devices, but only a common agreement from the majority of the manufacturers would lead the domain to an era where new technological possibilities could be utilized.

The current situation has a long history. Traditionally, development on BAS integration has concentrated mostly on the low level communication. Each manufacturer has focused on specifying communication protocols for their devices. These numerous and usually proprietary protocols have not allowed interoperability between devices from different manufacturers, or even between different products from the same manufacturer. To solve this, standardization of the building automation protocols started, first on a national level, and then on the international level [7]. Unfortunately, manufacturers promoted their own protocols for standards and the problem remained unresolved. This is why today there is a huge number of different competing standards for the device communication protocols.

As devices were communicating using various incompatible protocols, the only way to connect them under a common system was to implement protocol converters or gateways, which could communicate using multiple different protocols. These type of gateways, that are able to interconnect devices under a common system using device adapters, are called integration platforms. Figure 1 illustrates the architecture of a typical integration platform. At the lower level, the devices and subsystems are communicating using the specific field level protocols. Gateways are able to communicate with them and translate the device descriptions and controls to the common format used on the higher level backbone network. Then, a local control terminal can connect to the backbone network for managing the system. An additional gateway can also be used to connect the backbone network to the Internet allowing the usage of a remote control point.

More recent research has considered using IP networks as a backbone for the BASs. This has simplified the system architecture by allowing easier integration with the Internet and usage of the existing wired and wireless LAN cabling. As a consequence, new higher level building automation standards were developed. These standards are

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), June, 4–8, 2012, Valencia, Spain.

commonly based on the existing Web technologies, such as XML, SOAP and RESTful Web services, and are designed to be used with other techniques, such as WSDL descriptions and UDDI registry.

In complete independence from the BAS domain, research on Multi-Agent Systems (MAS) has focused on flexible mechanisms to coordinate networks of (usually distributed) autonomous computational entities. MAS' suitability to solve complex heterogeneous distributed tasks has been demonstrated in various applications[27]. MAS typically consist of a set of autonomous and intelligent agents operating in a certain environment sharing knowledge. The idea of such systems is that responsibilities are distributed and the system is managed through the cooperation of multiple agents. Each individual agent is capable of flexible interaction, and of reactive, pro-active, and social behavior. While the agents can request other agents to execute an action, the final behavioral decision remains within each agent. This cooperation and distribution features of MAS makes it an especially appropriate approach for implementing systems which are too complex to be managed by a unique, centralized control system.

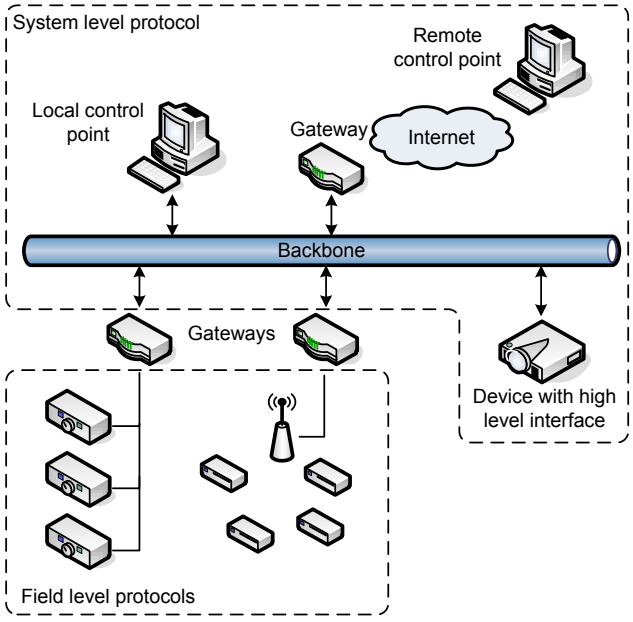

Figure 1: A typical building automation integration platform. [14]

Another research area has concentrated on the knowledge representation, more specifically, ontologies, to formally represent the concepts on a certain domain and their relationships. Software ontologies can be used by applications, *e.g.*, software agents, to reason about that domain. Particularly, ontologies differ from the commonly used metadata descriptions, controlled vocabularies, and XML Schema definitions in their support for advanced definitions for the relationships between the concepts. These relationships create a powerful way to describe the concepts and give them a semantic meaning.

In this paper, we propose to combine state-of-the-art Web technologies, MAS design, and ontologies to build an agent-oriented architecture for BAS. The proposed system can benefit from the strengths of these techniques and result in a more robust, distributed, and modular system design. To do so, we will analyze the approaches to that same solution which has been attempted previously, and argue how the emerging technologies can help overcome the problems those previous proposals had.

The rest of the paper is organized as follows: In §2 there is a description of the related work in the field, including also an overview of our previous approach. In §3 the proposal for the agent-oriented architecture for the BAS is presented in detail. §4 includes the description of a use case to which the solution is proposed to be applied. It also contains a detailed description of the proposed implementation and a discussion on the benefits of the new system. Finally, in §5 the conclusions are presented.

2. BACKGROUND

The problem we are tackling here requires choices at two different levels. On one hand there is the general architecture of the system, which states the elements involved and the communication paradigm used among them. On the other hand there is the internal functioning of each component, which eventually define the responsibilities and capabilities of each device. Eventually, both are strongly related, since the level of distribution in the system (defined by the architecture) will constrain the potential capabilities of each component. In this section, we will first review the solutions proposed in the bibliography to the architecture. Then, we will discuss the solutions proposed previously in our line of research to combine MAS with Web technologies, which has led us to our next step. Finally, we describe our earlier work done for Web services based BAS.

2.1 Approaches in literature

There are several solutions on how to organize the elements of a BAS. Those solutions are usually defined by the degree of distribution/centralization of its components. Most classical approaches use highly centralized architectures which gather most (if not all) responsibilities within a single, central component. An example of that is the solution proposed in [17]. In it, there are specific servers (called domoNetWS) which have the task to provide the Web services offered by various devices in their name. Those devices implement an interface which is used by the domoNetWS to forward the requests for their Web service methods. In [26] the same type of centralized architecture is used, one where the Web services are published by a main server instead of the actual device providing them. In all those cases, that server must somehow act as a middle-man, and all those additional communications must be handled.

Opposite to that centralized scheme, some proposals use a more distributed solution. There is a large degree of decentralization of components in such complex networks, but the main idea is to split responsibilities among the various entities. An example of that is [25], in which each device acts both as a server (publishing its Web services) and as a client (using the other device's Web services). To do so, a gateway has to be implemented which translates the device specific protocols into a common language (SOAP, in the case of [25]). This distributed paradigm of networking BAS had a big drawback due to technological constraints. Even though its goal is to avoid centralizing services, it required the use of a service repository within the network which would be

implemented as a UDDI registry [5]. As a result, this architecture is less centralized than the previous alternative, but still includes some components which cannot be distributed. In Section §3 new technological solutions will be described to avoid that restriction.

In the distributed solutions for BAS network communication, the responsibilities are highly spread in the network. Each device publishes its own services, and at the same time can query the rest of devices for their published services. This highly interconnected topology provides each component with a large amount of information. Information which potentially enables certain levels of complex behavior and, eventually, of smart interactions. It is in this context that MAS and Web service oriented networks meet.

There has been considerable research by the community on the integration of MAS and Web service technologies. The main problem in that integration have been the differences between their communication protocols [22]. In agent technologies there are a set of standards defined by IEEE-FIPA regarding how to describe knowledge and how to share it with each other. In Web service technologies, those standards are set by W3C/IETF. In both fields standards have been defined for communicating (ACL and SOAP), for description (DF-Agent-Description and WSDL) and for registering (DF and UDDI). Unfortunately, those standards are not compatible straight away.

A frequent solution for such problem is the development of proxy connectors between both technologies, engaging that way a translation process between both ends which eventually enables communication. WSDL2JADE[12], WSIGS[11] and WS2JADE[20] are all examples of that approach. These solutions try to combine all the research effort done along the years in these two separate areas in order to obtain the complete functionalities of each of those solutions without imposing restrictions on the other. That approach assumes a complete independence and a lack of overlap between both specifications. Agent platforms, however, were developed in isolation to network standards, which forced them to define their own paradigms. As a result, agent platforms specify not only the high-level description of agents, but also most of the low level communications protocols. Is in those features that agents research and Web services collide. However, as Web technologies evolve, so do the features they offer. In Section §3 we will describe the new possibilities that emerging technologies like WebSockets offer to avoid these problems.

2.2 Previous Work

Our previous approach [14] considered a Web services based integration platform. In contrast to the solutions that implement centralized server functionalities, our architecture consisted of a unique stand-alone device server, and device adapters and user agents as client applications. While a typical building automation gateway (Figure 2B) supports integrated protocol adapters and provides a separate high level interface, we proposed to simplify the architecture by using a higher level XML based protocol for communicating also directly with the devices. This way the system could avoid intermediate protocols and thus additional data conversions [14].

We selected to use open Building Information eXchange (oBIX) interface and data model in our system. The specification is rather simple and offers a RESTful Web services interface for the communication. The data model is extensible and based on XML. Because of this extensibility, we could easily implement the required additional functionalities and define our data models.

The system architecture is depicted in Figure 2A. A centralized server offers a Web services interface for the client applications. Devices can connect to the server directly, through a remote device adapter, or a local adapter. Thus, the devices can be located anywhere on the Internet regardless of the location of the server. As they only implement the client side functionality, they do not need to have a known IP address, but they might need to regularly poll the server for data updates.

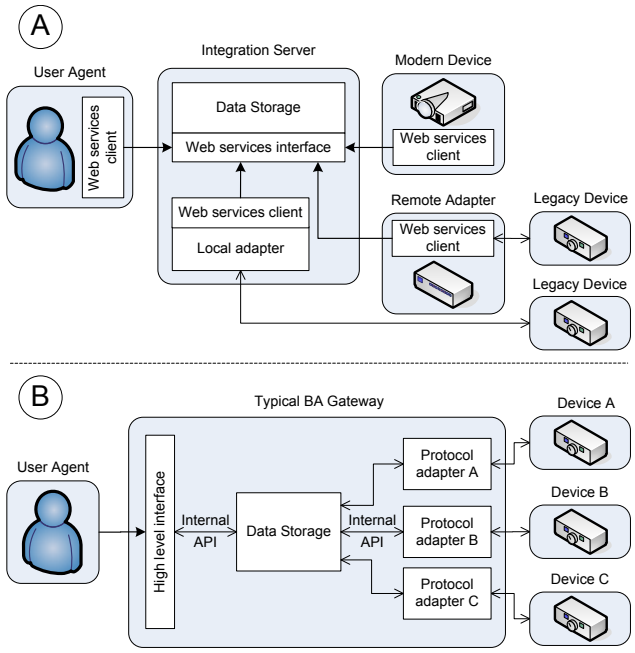

Figure 2: A: Starting point system architecture; B: A typical building automation gateway. [14]

The idea behind the architecture arose from the recent IP protocol development for small and simple devices with limited resources for computing. The IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) [18] allows to integrate IP stack even into very small battery-powered devices. Accordingly, the Constrained Application Protocol (CoAP) [23] is developed for the application layer to support the same purpose. Specifically, it is designed to easily translate to HTTP, thus supporting the vision of the Web of Things (WoT).

In the implemented system, the devices sign up to the server by publishing their data on it. To be notified of the changes in this data, they have to regularly poll the server. This type of communication is very suitable for simple devices, which only act as sensors and thus do not need to receive external commands. For the actuators instead, the communication schema is not as well suited. To reduce the additional network traffic overhead caused by the polling, we used a long polling technique (Figure 3), which is generally utilized in modern Web applications. When using this technique, the server holds the request and sends the response only when a new data update is available. In case

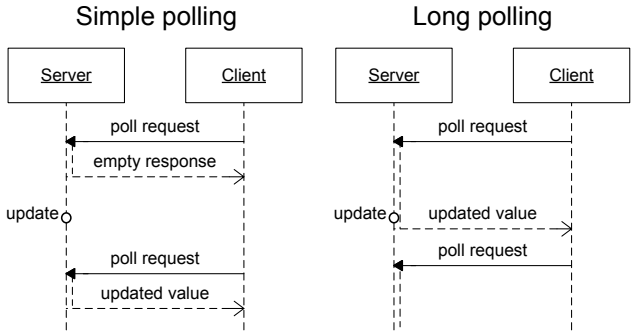

Figure 3: Simple polling compared to long polling technique. [14]

```
<obj is="smartApplication">
 <int name="stateChangesToLive"/>
 <bool name="active"/>
 <bool name="manualControl"/>
 <bool name="falseStateChange"/>
 <bool name="currentState"/>
 <reltime name="pollInterval"/>
 <op name="makeCondition" in="obix:Nil"
     out="condition"/>
 <op name="deleteCondition" in="obix:ref"
     out="obix:Nil"/>
 <op name="makeWatchState" in="obix:ref"
     out="watchState"/>
 <op name="deleteWatchState" in="obix:ref"
     out="obix:Nil"/>
 <list is="conditions" of="condition"/>
 <op name="makeAction" in="obix:Nil"
     out="action"/>
 <op name="deleteAction" in="obix:ref"
     out="obix:Nil"/>
 <list is="actions" of="action"/>
 <op name="delete" in="obix:Nil"
     out="obix:Nil"/>
</obj>
```

Figure 4: The smart application contract in oBIX.

the data on the server does not have frequent updates, this can dramatically decrease the network traffic.

To support the automatic control of the devices connected to the system, we implemented and integrated a rule engine into the server. An individual rule, called a *smart application*, had a set of properties to adjust its functioning, a list of conditions, and a list of actions (Figure 4). The management of the rules was possible using the oBIX Web services interface and the rules could directly command the devices based on their condition logic. The idea behind having only one common interface was to have a simple and modular system, where multiple control systems could function at the same time. Rules could depend on and control each other. The same way it could be possible for separate control systems to exploit the features of each other [15].

We demonstrated the integration of the devices into a modern BAS using the integration platform and the interoperability between the devices using the rule engine. We also demonstrated how individual *smart devices* can have some amount of logic integrated into them. Smart devices could be connected to the integration platform, but were also able to function autonomously and do reasoning based on their internal logic, and communicate directly with other devices, integration platforms, and rule engines.

3. ANALYSIS

3.1 Technological Analysis

HTTP is the most commonly used transfer protocol over the Internet. It was originally designed following the request-response paradigm. The main motivation for such design was to keep the server implementation simple (*i.e.*, the server is stateless and it does not store any client session state). Unfortunately, such simplicity also entails several limitations. One of such limitations is the inability to implement full bi-directional connections between server and client. Although there are some workarounds to implement asynchronous messages from the server to the client in HTTP (*e.g.*, Comet), those techniques usually require complex server implementations [16], and still cause the transmission of unnecessary data.

WebSockets[8] is a web communication protocol which provides bi-directional, full-duplex communication over TCP. The WebSocket protocol was standardized by the IETF on December 2011, and its API is being standardized by the W3C [13]. To obtain a WebSocket connection, first a standard HTTP connection has to be established. Then that connection is upgraded to a WebSocket connection through a handshake interaction in which the client request for the upgrade and the server replies. After that, the server and the client can exchange messages asynchronously until the connection (which is much like a bi-directional pipe) is closed. Once the connection is upgraded to WebSocket, the differentiation between Server and Client becomes virtually nonexistent, since both write and read (*i.e.*, send and receive) proactively. Through WebSockets, both text and binaries can be sent in either direction at the same time which improves real-time applications over TCP.

While in an established WebSocket connection there is no practical difference between the client and the server, they are both needed for creating the connection. Some approaches for direct browser to browser communication have been proposed [19][24], but in practice the server functionality is always required for establishing the connection. Thus, these approaches actually implement the server functions in the browser for this purpose.

WebSockets are particularly appropriate for real-time, event-driven applications, since a device with updated information can pro-actively notify the rest of the devices immediately. This idea is dramatically different from the Server/Client paradigm, in which there has to be a request from the client for the server to provide the data. In the scope of MAS, an agent can communicate with other agents when it considers appropriate to do so. Comparing the requirements of MAS and the features of WebSockets, it can be asserted that the spirit behind WebSockets fits better the MAS paradigm than a basic Server/Client communication protocol.

However, the benefits of WebSockets in networking are not limited to its underlying philosophy. Enabling full-duplex, bi-directional flows reduces significantly the network traffic and also the latency due to a reduction in the number of steps of communication (*i.e.*, less packages being sent).

Comparing the Comet HTTP-based solution to the WebSockets[13], WebSockets can provide a 500:1 reduction in unnecessary header traffic, and a 3:1 reduction in latency.

3.2 Architecture Analysis

The architecture we propose tries to enclose the strengths of the MAS and Web technologies, while avoiding past issues related with coherency. Building automation domain needs solutions for the interoperability between various different systems, and as discussed earlier, Web technologies have been proposed to be used for this kind of task. This approach brings forth a number of benefits as BASs can employ well established Web technologies and use the existing network infrastructure in the buildings. At the same time, MAS have shown their advantages in autonomous and intelligent behavior of the agents. FIPA has successfully developed and standardized the key technologies for building such systems. Multi-agent platforms have been implemented to support easy development of the agent-based solutions.

The problem of this kind of architecture is to define which components of each domain are employed. As mentioned above, FIPA has standardized numerous technologies for agent development. Thus, the new system should try to utilize these to as big extent as possible in order to maximize reuse. However, while we want to bring the architecture closer to the MAS, we also want to keep a tight relation with the Web technologies. As seen in the Figure 5, our architecture consists of FIPA compliant agent platform technology combined with the WebSocket communication between the agents. Other required communication techniques are derived from FIPA standards that are leaning towards the common Web standards. FIPA Agent Message Transport Protocol (MTP) for HTTP [2] is used to establish the communication between the agents. In our proposal, that connection will be upgraded to use WebSockets when required. At the same time, we can guarantee that the messages used by the agents to communicate with each other are standard FIPA-ACL messages at all times. A key feature in the proposed architecture is that of avoiding unnecessary translations between standards, since WebSockets do not make any restriction in the standard of the messages being sent.

In practice, beyond the inter-agent communication, each individual agent in the system can be connected to a real physical device, for example, an RFID reader or a pill dispenser. Thus, each agent that has a device connected to it needs a device interface module for the device specific input data. From the device-based input data, this device interface module uses an ontology alignment to translate the information from the low level device protocol to the corresponding representation in the Knowledge Base (KB). Naturally, it is also responsible of communicating the relevant changes in the KB back to the device in the native syntax of the device.

Following the MAS philosophy, we suggest that the nature of the device should define the purpose and the functionality of a specific agent. It is natural for an agent connected to a device to deal with the reasoning related to that device, and communicate with other agents to obtain and provide additional information about the environment if needed. An agent platform is harnessed for that purpose. Agent receives the device representation from the device interface module and thus does not need to care about the communication with the actual device. Most agent platforms can include a

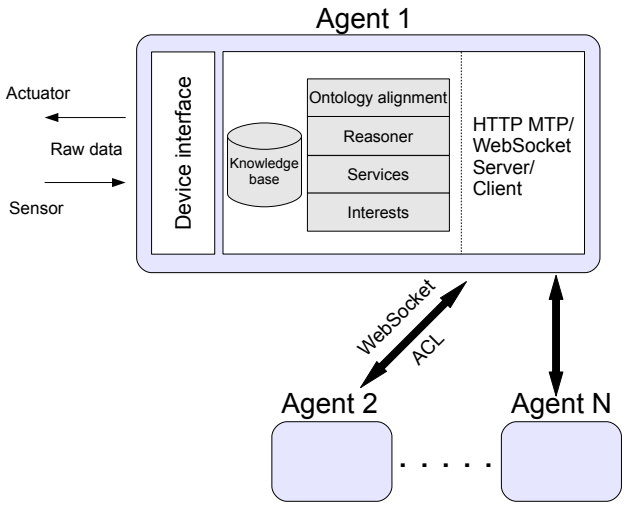

Figure 5: An agent-oriented architecture for building automation systems

reasoning module which allows its agents to reason about the environment based on the current world state represented by their KB. Alternatively, an agent platform can further communicate the world state to a remote reasoner service, which manages the intelligence in a more centralized manner. Although the addition of the WebSocket technology does not constrain the agent reasoning architecture being used, later on we will see how the distributed version fits better that approach.

For the social behavior, most agent platforms provide an HTTP MTP server and client, which can be used for establishing connections between agents [2]. Is in this point where we propose the addition of a WebSocket to be implemented and integrated into the platform. Since WebSocket connections are based on HTTP, that module would be in charge of upgrading HTTP connections to WebSocket when necessary. As mentioned earlier, the WebSockets can be used to enable P2P bi-directional connections between the agents while allowing ACL as standard communication language. That integration would provide every agent with the capability of creating WebSocket connections with any other agent within HTTP connectivity range.

The most relevant effect of introducing WebSocket technology into MAS involves a change in each agent's behavior. Given that WebSockets enable server-push actions, agents no longer need to periodically request other agents for data. By using a type of subscription service (not to be confused with Web services), agents can be automatically and immediately updated with any relevant information any other agent may have. Once an agent subscribes to another agent's service, a WebSocket connection is established between these agents, and both can send data through it. Accordingly, even though an agent can count on being kept updated, it can still request for an update through that same WebSocket. That exceptional request can be used for safety and coherency reasons since, for example, an agent may want to make sure its KB is correctly updated with the latest knowledge and that no data was lost in the communication when some unexpected state is reached.

The proposed architecture forms a MAS where autono-

mous intelligent agents can deal with the tasks relevant to their purpose and capabilities. The system promotes distributed reasoning, but does not exclude the possibility of using remote reasoning services. It takes advantage of the standard FIPA compliant agent platform technologies and combines them with the bi-directional WebSocket communication paradigm. Next we detail the particularities and possibilities this technology introduces in the registry of agents.

3.2.1 Directory Facilitator

Standard FIPA architecture offers a solution for the service discovery with the Directory Facilitator (DF). The DF is designed to offer a centralized registry for the service discovery in an agent platform. It is capable of detecting new agents in a platform and adding them to the registry. It contains DFAgentDescriptions (DFDs), information of what services agents are offering and where to find them [1]. Thus, when any agent need to use a specific service, the DF can help the agent to localize it. The agent can then at any time request the service using the location once provided by the DF.

With the proposed use of WebSockets for the communication, a subscription mechanism for the services is desirable. Since agents no longer need to request service providers for data, it is necessary that those service providers implement mechanisms to keep the rest of the agents within the MAS informed. The Java Agent DEvelopment Framework (JADE) [4] offers its own implemented DF Subscription Service for agents, based on FIPA's Subscribe Interaction Protocol (FSIP) [3]. With this service, agents are able to register their interest in certain services in the DF. Once an agent has shared its desire for specific services (which we will call an agent's *Interests*), the DF will automatically serve to the agent the information regarding where to find them and will keep the agent updated if new services are published. This DF Subscription Service however does not specify subscribing policies among individual agents (*i.e.*, without one of them being the DF). Such specification is now required in order to use WebSockets. Service providers will be the ones responsible to keep the rest of the agents updated, and subscription among agents will be the cornerstone of communication.

To define the required specification we will consider the existing standard. As said before, FIPA has its own subscription standard for MAS (FSIP) but it only covers the few low level steps between the request for subscription and the reply from the service provider. These steps do not fully define the requirements of the new paradigm, which is based on subscriptive actions at a higher level. It is therefore necessary to extend it in order to specify all needs of WebSockets. Next we describe the basic steps needed to be added in this new scenario (Figure 6), both in the cases where a new agent is added to the MAS and when that agent is already within it. For the steps which completely overlap with FSIP we will refer to it for details:

1. A new agent *Agent 1* connects to the MAS through the DF.
2. The DF adds the services (DFD) of *Agent 1* to the registry.
3. *Agent 1* sends its interests to the DF using DF Subscription Services.
4. The DF replies with a list of services matching *Agent 1*'s interests.
5. *Agent 1* requests WebSocket connections to the agents offering services that it is interested in.
6. Once the WebSocket connection is established with those agents, *Agent 1* requests subscriptions for these services through the WebSockets (FSIP).

After performing these six initial steps, the agent is fully established within the MAS. From that point on, there are two additional scenarios in which an agent needs to perform subscription related actions; for subscribing to new services that have appeared, and to accept subscribers to its own services. In that regard, those actions could take place at any time and any number of times once the agent is fully established within the MAS.

First we describe the steps required for subscribing to new services.

- *Agent 1* receives a subscription response message from the DF with a list of new services which match *Agent 1*'s interests.
- *Agent 1* requests WebSocket connections to the agents offering services that it is interested in.
- *Agent 1* requests subscriptions for these services through the established WebSockets (FSIP).

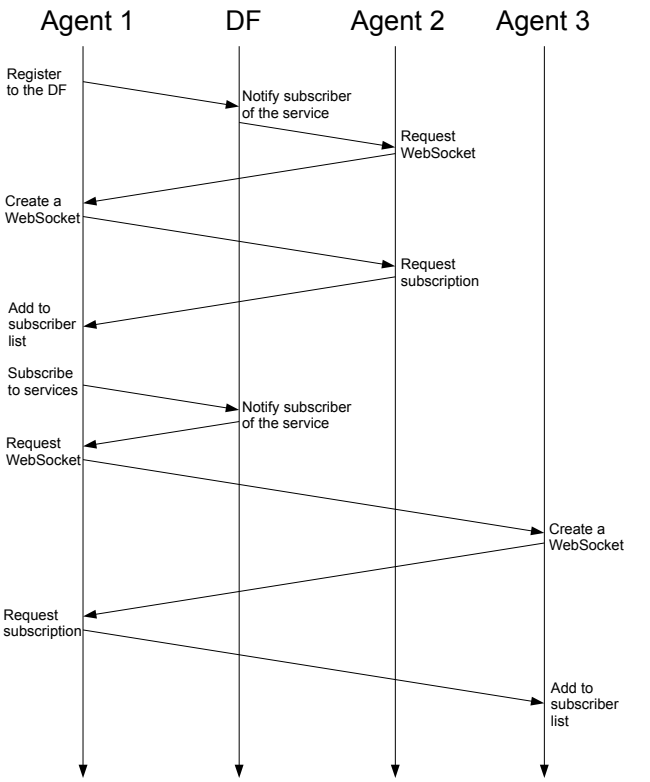

Figure 6: An agent registering to the DF.

The second scenario takes place when a different agent wants to subscribe to one or more services provided by *Agent*

1 itself. Obviously such request can happen at any time during the lifetime of a service providing agent. It is notable however, that these steps may happen just after the agent connects to the MAS, since interests could already be defined within the DF by the time the agent registers its services into it. The first step is omitted if agents in question have already established a WebSocket connection between them.

- *Agent 1* receives a request to create a WebSocket connection with another agent and creates it.
- *Agent 1* receives a subscription request for some of its services from the agent through the WebSocket (FSIP).
- *Agent 1* adds that agent into lists of subscribers of the corresponding services.

Following these steps in both scenarios ensures that agents are subscribed to all the services they are interested in, and at the same time, that they are accordingly providing their own services to the other agents. Notice how, at all times, the DF keeps an updated list of services and interests for the whole MAS.

3.2.2 Subscription management

As a result of the changes required by the use of WebSockets, there is a need to keep a *List of Subscribers* in each service providing agent. Considering that service providing agents are the ones responsible to communicate any update to its subscribers, each of those agents must keep an updated list for that purpose (Figure 7). Such list must associate each service provided by the agent with a list of agents subscribed. Moreover, for each of those subscribed agents there must be a mapping to the underlying WebSocket connection.

Similarly, it could be useful for the agent's internal logic to have an updated *List of Subscriptions*. Such list would relate data inflow requirements (*e.g.*, what is the current temperature, who is currently at home, etc.) with running WebSockets through which that data would be expected to arrive. That list could be used to implement update listeners and to manage the existing subscriptions. An overview of the components of that list can be seen in Figure 7.

By using the List of Subscribers to direct all server push communications, we can guarantee that those and only those agents which are interested in a service will be kept updated regarding that service. Notice also how a broadcast entry could be added to the List of Subscribers, which would include all the agents within the MAS.

Once the WebSocket connections are established following the subscription requests, no more request or polling is required. Service providers will push data updates to their subscribers with each new update. This will radically reduce the network traffic in constantly changing environments (*i.e.*, with many agents connecting and disconnecting from the system). Agents no longer have to periodically query the DF to keep an updated list of available services. When a new agent comes along, the needed subscriptions are automatically performed following the steps described earlier. Furthermore, if an agent was to change its services or interests, it has only to notify the DF and to cancel any related subscriptions. Also, this solution supports easy failure recovery, as after a reboot an agent can follow the initialization steps to recover its own subscriptions and to provide its services to other agents based on the List of Subscribers that it holds.

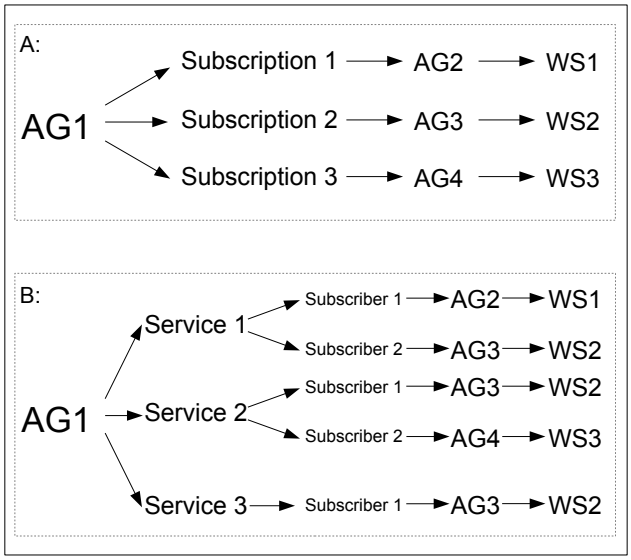

Figure 7: A: List of Subscriptions; B: List of Subscribers.

4. DISCUSSION

4.1 Use Case

From the analysis performed in the previous section, we intend to implement a system based on it. Concretely, we will implement the use case outlined in [10]. That use case tried to tackle the difficulties faced by the elderly people living on their own with the use of BAS. The use case considers the problem of supplying medication to someone with reduced mobility while enabling monitoring of that person's compliance to the pharmaceutical treatment. The non-compliance with those medical indications is considered as a major drawback nowadays [9].

The use case includes devices such as a domotic door, a pill dispenser, motion sensors and temperature sensors. These devices naturally reside in different physical locations. While those could be connected using a centralized integration platform, distributed designs offers a more robust solution. To further decrease possible bottlenecks, the functionalities of the system can thus be distributed following the MAS paradigm. In that case each agent could handle the tasks that are relevant or related to its own device.

An additional feature provided by the proposed architecture is that of enabling P2P connections and server-push communications. As such, each autonomous agent can proactively notify the rest of the MAS about the changes it detects. In this scenario, let's consider the following situation:

The domotic door detects that the user has just arrived home. Pro-actively, it communicates that fact to the rest of the agents within the MAS which are interested in it. When the pill dispenser learns that the user is at home, it deliberates about whether there is a medication dose that must be delivered to her. Again, this update is immediately communicated to the interested agents. The interface, which is

an autonomous agent interested in facts which require interaction with the user, notifies her that there is a medication dose ready. If the medication dose is not taken, that may suppose a health risk for her. Depending on that risk, the pill dispenser may eventually notify the alarm agent about it. In turn, if the risk is *high* enough, (*e.g.*, importance of the medication, number of doses not taken, etc.) the alarm agent will communicate with the responsible person (*e.g.*, doctor, caretaker, insurance company, family member, etc.) for preventive actions.

Although this situation can be implemented following classic communication standards like Web services, WebSockets allow it to be much faster and lighter due to a reduction of traffic and the pro-activeness of agents. In this example it becomes obvious that in many situations the behavior of agents within a MAS is supposed to be pro-active. That is so because the rest of the MAS is interested in being kept updated all the time. The benefits of WebSockets are especially relevant for the use case, where each agent is interested in only a piece of the whole domain of facts (*e.g.*, access and presence in the house, medical treatments, emergency procedures, etc.), and where latency reduction is a priority (*e.g.*, medication overdoses can require rapid responses).

This change of paradigm (from Request/Response to server-push), in our opinion, will change the way intelligent agents are implemented. Given that client requests are no longer the basis of all communication, agents can trust the pro-activeness of other agents, and expect to be kept updated by them when relevant events occur. Although in certain cases agents may want or need to actively request for data from other agents (*e.g.*, a new agent is added to the MAS and needs to build its KB, there has been some unexpected behavior and an agent want to make sure its KB is coherent with the current state of the world, etc.), we consider that such cases will be a small percentage of the total number of interactions among agents. Meaning that the gain in efficiency and latency by using WebSockets is relevant enough to apply it.

4.2 Technical discussion

The motivation behind the work presented here has already been outlined in previous sections. The goal is to propose an architecture of a system capable of using state-of-the-art agent technologies in order to benefit from all the research done in the field. Nowadays, when developing agent-related systems it is recommended to follow the standards defined by FIPA, since they compose a set of agreed and tested protocols. By doing that, one can add and integrate most available platforms and libraries related with agents without much effort.

The other main point of our proposal is the use of WebSockets. WebSockets represents a significant improvement in network communications through true bi-directional, full-duplex flow of data. The use of the WebSockets protocol to manage MAS communications is appropriate at various levels. First of all, it satisfies the pro-activeness and independence requirements found in the essence of MAS. At a lower level, WebSockets define a unique communication pipe between two peers (*i.e.*, a socket) for both sending and receiving messages between them. That unification simplifies significantly the communication methodology between agents. Agents located in heterogeneous networks would especially benefit from that feature, since the use of firewalls and proxies would be simplified.

By avoiding any type of polling, the number of packages needed to be sent in order to establish and maintain a communication is reduced significantly. Connections do not need to be established each time a new request for data is desired, and, given that an agent can expect to be automatically informed by another one once its interest has been stated, there is no need to request messages at all. The WebSocket connecting two agents is kept open until one decides to close it, which would happen only when one agent is no longer interested in communicating with another.

Similarly to P2P networks, agents in our system act both as suppliers and consumers of each others resources and thus the workload of the system is distributed among the agents. To tackle the scalability issues, today's P2P networks also administer the discovery mechanisms in a distributed manner using supernodes or distributed hash tables [21]. To further increase the robustness of our system following the P2P principles, alternative solutions, such as integration of Distributed Directory Facilitator [6] could be used. That will be one of our future lines of work.

Using WebSockets for the communication also allows easier integration with the traditional Web applications. The huge success of the Web has resulted in development of easy tools for implementing browser based applications. Numerous techniques from small libraries to complete frameworks are created and new emerging technologies (*e.g.*, WebSockets) are already included in the newest tools. This Web convergence with agent platforms thus allows the usage of numerous Web tools for developing the Web front end for managing the building automation system, or any other system utilizing the Web technologies.

Finally, the proposed architecture makes use of several protocols without binding the system to a given structure. In our particular case, we intend to develop further into a distributed architecture (see §2) since we consider it the most appropriate structure in order to maximize the benefits of the technologies being used. However, the flexibility of both Agent Platforms and WebSockets enables the possibility of using other, strongly centralized schemes. Considering that WebSockets is a communication protocol with no restrictions whatsoever regarding the data being sent (it can be either text or binary), and that Intelligent Agents can vary in size depending on the case, the topology of the network could be adapted. There could be the case, for example, in which a few centralized agents are defined which connect through WebSockets with some *dummy* components. The centralized agents would have open connections with the dummy components to be updated of any change in their knowledge of the world. In that case, those dummy components would be simple interfaces between any sensor or actuator and the Intelligent Agents.

5. CONCLUSIONS

We have proposed to combine state-of-the-art Web technologies and MAS design to build an agent-oriented architecture for BAS. The proposed system will benefit from the strengths of these techniques and result in a more robust, distributed, and modular system design. We have analyzed various approaches for the same solution which have been attempted previously, and argue that the emerging technologies, WebSockets in particular, can help to overcome the problems of the previous solutions.

The main innovation of the proposed architecture is the integration of WebSocket communication protocol into the existing Agent Platforms, for the communication between the agents. The benefits of that protocol when compared to the typical solutions used (*e.g.*, Web services), makes it an interesting approach to test. WebSockets reduce the amount of traffic as well as the latency in most communications. WebSocket communication protocol provides bi-directional, asynchronous full-duplex communication over TCP, which is well suited for the agent communication using ACL, because it is completely agnostic for the message payload. At the same time, this freedom that WebSockets provide from standards, allows the final architectural design of our system to be mutable in many ways. It does not require translating methodologies as most proposed solutions do (*e.g.*, translating from ACL to SOAP), and messages can be kept in ACL at all times. At the same time, the fact that WebSockets define restrictions only at low communication levels allows this kind of architecture to be used in various configurations. As an example, both highly distributed and highly centralized MAS would benefit from the use of WebSockets.

To motivate the proposed architecture, we have presented a health care related use case, where the suggested MAS will be implemented. The architecture can be easily applied for the use case for a proof of concept implementation. The distributed architecture seems to naturally fit the scenario of a MAS, where the devices are responsible for the reasoning most related to their own functioning and purpose, and subscribe to additional information to be received from other agents. In the critical use case of Assistive Technologies, the potential benefits of such application become extremely relevant and justifies this research line.

6. ACKNOWLEDGMENTS

Part of the work was done as part of TIVIT Devices and Interoperability Ecosystem (DIEM) project Building Automation work package and supported by The Doctoral Programme in the Built Environment (RYM-TO) funded through the Academy of Finland and the Ministry of Education and Culture. Authors also want to thank Andrey Litvinov for his contribution for the earlier work done in developing the oBIX solutions that built basis for the current work. This work has also been partially supported by the SUPERHUB project (SUstainable and PERsuasive Human Users moBility in future cities - Co-financed by the European Commission - ICT-FP7-289067) and the IWALKER project (Collaborative Rehabilitation, TEC2011-29106-C02-02). We would also like to thank Javier Vázquez-Salceda and Ignasi Gómez-Sebastià for their useful comments during this work.

7. REFERENCES

[1] *FIPA Agent Management Specification*. Foundation for Intelligent Physical Agents. 2000.

[2] *FIPA Agent Message Transport Protocol for HTTP Specification*. Foundation for Intelligent Physical Agents. Mar. 2002.

[3] *FIPA Subscribe Interaction Protocol Specification*. Foundation for Intelligent Physical Agents. 2002.

[4] F. Bellifemine, G. Caire, and D. Greenwood. *Developing multi-agent systems with JADE*, volume 5. Wiley, 2007.

[5] T. Bellwood, L. Clement, D. Ehnebuske, A. Hately, M. Hondo, Y. Husband, K. Januszewski, S. Lee, B. McKee, J. Munter, et al. UDDI Version 3.0, 2002.

[6] C. Campo. Distributed directory facilitator: A proposal for the fipa ad-hoc first cft. 2002.

[7] M. Felser and T. Sauter. The fieldbus war: History or short break between battles. In *IEEE WFCS*, pages 73–80, 2002.

[8] I. Fette and A. Melnikov. *The WebSocket Protocol*. IETF HyBi Working Group. 2011.

[9] I. Gómez-Sebastià, D. Garcia-Gasulla, and S. Alvarez-Napago. Society of situated agents for adaptable eldercare. *ERCIM news*, (87):23–24, October 2011.

[10] I. Gómez-Sebastià, D. Garcia-Gasulla, C. Barruè, J. Vázquez-Salceda, and U. Cortés. Alive meets share-it : An agent-oriented solution to model organisational and normative requirements in assistive technologies. In *VI Workshop on Agents Applied in Health Care (A2HC)*, Casablanca, Morocco, December 2010.

[11] D. Greenwood and M. Calisti. Engineering web service - agent integration. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, volume 2, pages 1918–1925 vol.2, oct. 2004.

[12] D. Greenwood, M. Lyell, A. Mallya, and H. Suguri. The ieee fipa approach to integrating software agents and web services. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, AAMAS '07, pages 276:1–276:7, New York, NY, USA, 2007. ACM.

[13] I. Hickson. *The WebSocket API*. W3C, Editor's Draft. 2012.

[14] H. Järvinen, A. Litvinov, and P. Vuorimaa. Integration platform for home and building automation systems. In *Consumer Communications and Networking Conference (CCNC)*, pages 292–296. IEEE, 2011.

[15] H. Järvinen and P. Vuorimaa. Anticipatory lighting in smart building. In *Consumer Communications and Networking Conference (CCNC)*. IEEE, 2012.

[16] S. Loreto, P. Saint-Andre, S. Salsano, and G. Wilkins. *Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP*. RFC 6202. Apr. 2011.

[17] V. Miori, D. Russo, and M. Aliberti. Domotic technologies incompatibility becomes user transparent. *Commun. ACM*, 53(1):153–157, 2010.

[18] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. *Transmission of IPv6 packets over IEEE 802.15.4 networks*. Internet Engineering Task Force RFC-4944. 2007.

[19] M. Morris-Pearce. A secure p2p server in the web browser.

[20] X. Nguyen and R. Kowalczyk. Ws2jade: Integrating web service with jade agents. In J. Huang, R. Kowalczyk, Z. Maamar, D. Martin, I. Müller, S. Stoutenburg, and K. Sycara, editors, *Service-Oriented Computing: Agents, Semantics, and Engineering*, volume 4504 of *Lecture Notes in Computer Science*, pages 147–159. Springer Berlin / Heidelberg, 2007. 10.1007/978-3-540-72619-7_11.

[21] S. Ratnasamy, I. Stoica, and S. Shenker. Routing algorithms for dhts: Some open questions. *Peer-to-Peer Systems*, pages 45–52, 2002.

[22] M. O. Shafiq, A. Ali, H. F. Ahmad, and H. Suguri. Agentweb gateway - a middleware for dynamic integration of multi agent system and web services framework. In *Enabling Technologies: Infrastructure for Collaborative Enterprise, 2005. 14th IEEE International Workshops on*, pages 267–268, June 2005.

[23] Z. Shelby, K. Hartke, C. Bormann, and B. Frank. Constrained Application Protocol (CoAP) draft-ietf-core-coap-09. 2012.

[24] T. Ruottu and K. Markus. BrowserSocket.

[25] E. Tokunaga, H. Ishikawa, M. Kurahashi, Y. Morimoto, and T. Nakajima. A framework for connecting home computing middleware. In *ICDCSW*, pages 765–770, Washington, DC, USA, 2002. IEEE Computer Society.

[26] S. Wang, Z. Xu, J. Cao, and J. Zhang. A middleware for web service-enabled integration and interoperation of intelligent building systems. *Automation in Construction*, 16(1):112 – 121, 2007. CAAD Futures, 2005.

[27] S. Wilmott, J. Dale, B. Burg, P. Charlton, and P. O'Brien. Agentcities: A Worldwide Open Agent Network. *AgentLink News*, Issue 8, November 2001.