A Multi Agent Approach for the Representation and Execution of Medical Protocols

Armando Robles [1] and **Pablo Noriega** [2] and **Michael Luck**[3] and **Francisco Cantú** [4] and **Francisco Rodríguez** [5]

Abstract. This paper reports on our progress towards a framework for enabling intelligent organizations using MAS technology. Namely, the first stages of a bottom-up approach to the implementation of the framework. This time we discuss how it can be used for defining and executing medical protocols and show how we are applying the framework to implement an outpatient care medical protocol. We show how an outpatient care protocol, currently operational, has been agentified and is controlled by an organization middleware that is a preliminary version of the organization engine we proposed as part of our framework. The organization middleware reads workflow scripts at run-time and interprets them delegating tasks to specialized server agents that manage the access to medical records and business rules. These server agents in turn, communicate with specialized user agents that facilitate human interactions through traditional plain and grid forms.

1 Introduction

In the medical domain, the main focus of business process modelling is the modelling of *medical protocols* (MP). There are several languages developed for this type of modelling, with Glif (rule–based), Proforma (logic–based), Asbru (task–based) and Guide (Petri nets–based)[8] being good representatives of the state of the art in the field.[6]

While all such protocol modelling languages have their own advantages, we are concerned with capturing the *functionality* of MP *modelling* as well as the integration of the modeled MP as atomic procedures into a coherent medical information system. The key problem to be solved is the abstract representation of a *business process* in the *medical domain* and its on-line execution in a hospital. Hence, in addition to the modelled MP, we need an interpreter that reads the representation of the MP and executes and monitors the represented processes, which are, in fact, part of the system that operates in a distributed environment. We propose to address this problem within a general framework that unifies a high level description of how an organization should operate with its actual operation [9]. The framework is based on an institutional view of organizations and is built around an organization engine that generalizes the notion of electronic institution (EI) and the corresponding tools developed at IIIA [1].

[1] IIIA - Artificial Intelligence Research Institute, CSIC - Spanish Scientific Research Council, Barcelona, Spain, email:arobles@iiia.csic.es
[2] IIIA - Artificial Intelligence Research Institute, CSIC - Spanish Scientific Research Council, Barcelona, Spain, email:pablo@iiia.csic.es
[3] Electronics and Computer Science, University of Southampton, UK, email:mml@ecs.soton.ac.uk
[4] Tecnológico de Monterrey, México, email:fcantu@itesm.mx
[5] TCA Research Group, Monterrey México, email:frodriguez@grupotca.com
[6] *www.openclinical.org*

In the work reported in this paper we take a bottom-up approach to the design and implementation of the framework and focus our efforts on the agentification of application domain components (data bases, business rule repositories, workflow scripts, user interaction devices) and on the implementation of business process protocols in the context of an *outpatient care system*. This outpatient care system is a subsystem of a large hospital information system currently in operation. The system was developed by the TCA group, a medium-sized IT company whose business is the design, development and implementation of integral information systems. The outpatient care subsystem we report is currently in operation and is an agentified version of the original subsystem. TCA aims to continue with the agentification of the hospital information system and its other products along the lines described in this paper.

The implementation of the outpatient care protocol discussed here is built around organization middleware that is a preliminary implementation of the organization engine we are developing for the framework. This middleware sequences the invocation of business rules according to a run-time interpretation of workflow scripts and controls two types of agents that intervene in any given process: *server agents* are specialized in system components like business rules and medical records repositories, and *user agents* handle interaction devices to communicate with human users and external processes.

The contributions of this paper are: (i) a proof of concept for an agent-based implementation –in operation– for the outpatient care protocols of the hospital information system, (ii) organization middleware that includes a workflow engine and a grounding language, and (iii) a collection of user and server agents that interface the organization middleware with the components of a hospital information system.

The paper is organized as follows. Section 2, provides a quick account of organizations and electronic institutions, and an overview and the protocol for the outpatient care system. In Section 3 we outline the implemented architecture and report results of the implementation. In the last two sections, we discuss related work and future work.

2 Background

2.1 Organizations and Electronic Institutions

An organization (or a firm) is in essence a group of individuals that pursue their collective or shared goals by interacting in accordance with some shared conventions, and using their available resources as best they can [5, 7, 2].

Hospitals and other types of organizations have conventions that *organize* their activity in consistent ways so that employees and clients have some certainty about what is expected of them and what to expect from interacting with members of the organization.

A traditional institution is a means to organize, articulate, or in some other way structure human interactions. An *EI* is the computational counterpart of a traditional institution. Thus, while a traditional institution is a set of conventions that a group of humans follow in order to accomplish some socially agreed objectives, an *EI* is an implementation of conventions that apply to the interactions of agents that may be human or software agents.

An *EI* —as defined in the IIIA [6, 11, 4]— is specified through a *dialogical framework* that defines ontology and language conventions, and through a deontological component that establishes the pragmatics of admissible illocutory actions. This deontic component is currently operationalized by two constructs: (i) a *performative structure* that constitutes a network of scenes linked by transitions between scenes, where scenes are role-based interaction protocols and transitions describe the role–flow policies between scenes; and (ii) *rules of behavior* that establish role-based conventions for commitment making and satisfaction, and are currently expressed as pre and post-conditions of the illocutions admissible by the performative structure of the *EI*. The IIIA group has also developed a suite of tools (EIDE) to specify and implement electronic institutions [1].

Although it is not unusual to identify an institution with the organization that puts the conventions into practice, we want to make a clear separation of both concepts. We refer to the institutional aspects of an organization as being those conventions that prescribe the way in which the organization is supposed to function. These conventions may take the form of protocols and business processes, but also directives and other decision-making criteria and knowledge repositories that intervene in the day-to-day operation of the organization. We have proposed a framework to enable those institutional aspects in the enterprise information system of an organization [9]. This framework's architecture consists of three layers (as depicted in Figure 3). The top one is the specification of the institutional aspects that control agent (and human) interactions in the organization. The middle layer is what we call an *organization engine* that consists of an electronic institution that enacts the specification and middleware that — through what we call a *grounding language*— interprets the institutional actions as processes and transactions that take place in the bottom layer. The bottom layer contains the typical components of a traditional information system plus two types of agents that act as front-ends of the components (server agents) and of the users of the system (user agents). It is beyond the scope of this paper to discuss the framework further, except to say that here we are concerned mainly with the connection between the bottom and middle layers For this purpose we instantiate performative structures as workflows of the medical protocols, interpret patients and hospital staff as user agents, and control illocutionary actions involved in the workflows as actions in the operational hospital information system where user and server agents intervene.

2.2 Outpatient Care System

The outpatient care system is part of a larger system for the management and operation of hospitals, and involves a patient making appointments through the internet or through a call center, with the system giving an appointment according to availability. Once an appointment is made, the system embodies two main functionalities and a few subordinate ones. In the *patient consultation function*, the physician checks the electronic medical records (EMR), updates relevant information such as clinical history or diagnosis, and writes

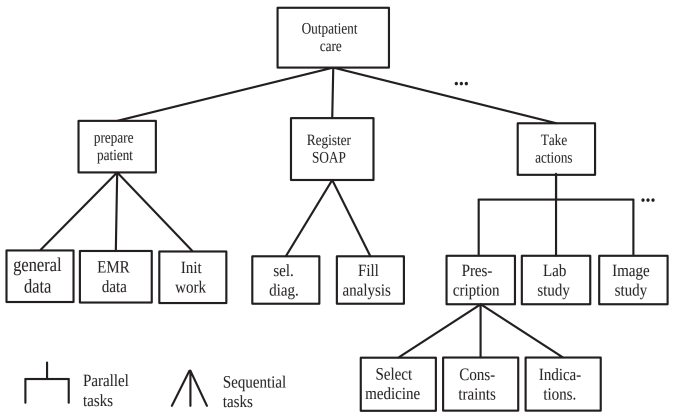

Figure 1. Part of an outpatient care medical protocol

prescriptions or takes referral actions, such as making lab appointments, hospitalization orders, etc. In the *patient services function*, the system directs the patient to obtain the medication and services prescribed.

To update the patient's file, the doctor has access to the patient's EMR with the individual's historical information. This information is indexed by features like date, incident or topic and includes, among others, treatment histories, diagnoses, laboratory results, image interpretations, nurses' notes, food-intake and diets, prescription history, blood transfusion requests, anesthetics records, etc.

The system also includes also the following functionalities: (i) it requests doctors to fill the proper medical forms and express applicable criteria after a visit; (ii) it allows doctors to register prescriptions, indications and diagnoses; (iii) based on those inputs the system matches the coding of diagnoses and therapeutic procedures in order to assign procedure codes; and (iv) since all these functions are cost-based, the systems uses the data for invoicing and reimbursement of medical fees.

2.3 Outpatient Care Medical Protocol

Figure 1 shows part of the medical protocol for outpatient care. After selecting the current patient from a list, the physician interviews him. Meanwhile, the system uploads all of its data (previous visits, allergies, patient identification, etc.). Upon completion of the interview, the physician registers the current patient information in a form, following the SOAP method (Subjective, Objective, Analysis and Plan).

Note that the standardized reporting forms, in addition to supporting recording of care data, enforce its correct retrieval through a systematic process that involves the following steps.

1. After registering the subjective (patient referred data) and the objective (such as data found in auscultation) contents of the visit, a preliminary diagnosis is made and matched against a list of codes. Interview analysis must be done in accordance with the diagnosis code.
2. Given this preliminary diagnosis, a prescription can be issued, although it is restricted by several factors:
   (a) Physician's specialty (i.e. a gynecologist can't prescribe psychiatric drugs).

(b) Patient's allergies and current conditions, such as pregnancy, ulcers, etc.

3. The physician proceeds to give the proper indications to the patient.
4. The preliminary diagnosis serves as a guide for requesting laboratory, imaging and other studies obtaining a final diagnosis. (i.e. every time that a bone fracture is reported, an X-Ray study is required).
5. Depending on the diagnosis, the patient is assigned a destination, namely actual in-hospital treatment, further ambulatory care or discharged.

After the prescriptions have been made and studies requested, the correspondent departments are notified. For example, if a Hematic Biometry is required, the Lab receives a test request for the patient (via HL7 [7]) and sets an appointment for performing the test, i.e. taking samples, etc.

When all the data and involved actions have been registered, the physician must fill the Treatment Plan forms (this may be done automatically by pressing its corresponding button).

Once all these steps are through, there are still a few more actions to perform (not shown in the figure):

1. Issue a sick-leave certificate which in turn triggers a new workflow script for filling this form.
2. Register the subsequent destination of the patient, and follow the corresponding MP.

(a) E.R., which means immediate action must be taken and triggers a new MP.

(b) Hospitalization. This triggers yet another MP for bed reservation, frontal page creation, data collecting, etc.

(c) Inter-consultantships, which restarts this MP with a new physician / specialty combination.

(d) Clinical history update

Finally, the physician can schedule a subsequent visit (Next appointment), triggering the scheduling agent and / or mark this visit as finalized (End Visit Button). This option, verifies that every bit of required data is completed and executes a final consistency check, then proceeds to store everything in the patient's permanent EMR.

3 An Integrated Environment for the Execution of Medical Protocols

Our aim is to integrate a typical information system for the management and operation of a hospital with a prescriptive counterpart, that should be capable of interpreting and executing medical protocols, and whose execution and monitoring involves the scheduling of tasks and resource allocations required by the intervening processes.

Thus, the outpatient care system described above is implemented in a multi agent framework, as shown in Figure 2. The middleware supervises actual domain components, such as business rules and data base components, represented by (Bag) and (Dag) *Server Agents* respectively. They handle all specialized tasks to serve the requirements of human users represented by *User Agents* (Uag). The middleware also reads workflow scripts from a respository, and these scripts are interpreted by a workflow engine that guides the actual execution of the system. Thus, the organization middleware, directed by the specified workflow script, integrates users and domain components.

[7] Health Level Seven (**HL7**) is one of several ANSI-accredited standards operating in the health-care arena. The domain is clinical and administrative data. One of its goals is to provide standards for the exchange, management and integration of data that support clinical patient care and the management for inter-operability between health-care information systems [3].

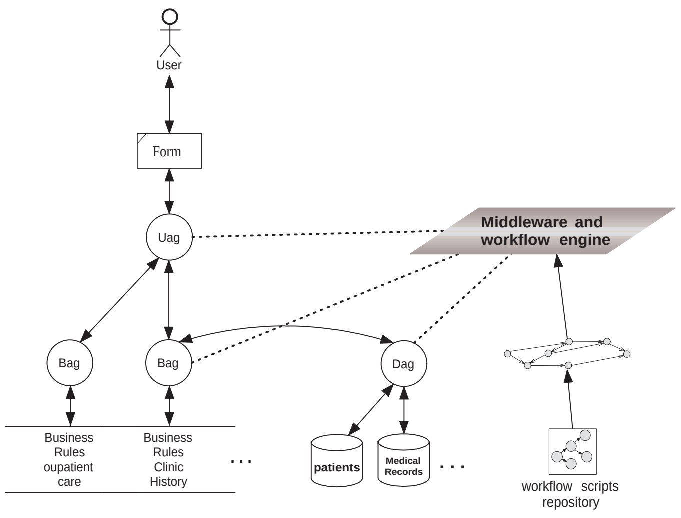
User
Form
Uag
Middleware and workflow engine
Bag
Bag
Dag
Business Rules oupatient care
Business Rules Clinic History
...
patients
Medical Records
...
workflow scripts repository

Figure 2. Integrated environment for the execution of workflows

3.1 The Middleware and Workflow Engine Layer

The organization middleware runs the system by placing *business domain* elements and users in contact according to workflow scripts. The basic functions of the middleware are:

- to log users into the organization, controlling user roles, agent resources and security issues;
- to monitor user interaction; and
- to load and interpret *workflow scripts*.

The workflow engine (*WFE*) has two components: the workflow specification language and the workflow interpreter.

3.1.1 Workflow Specification Language

The MP is defined as the proper interaction sequence of the domain components. Each workflow specification is stored in a repository as a workflow script. Since each domain component is represented in the environment by a specialized *server agent*, we have implemented commands for sending requests to the corresponding *server agents* for the execution of business rules, data base access, reports definition, and end user interaction.

Each task specified in the protocol (see Figure 1) is implemented as one of the following domain components:

- a business rule that could be from a single simple computation to a complete computer program;
- a data base access to add, delete, modify or retrieve information from a data base;
- a user interaction through a specialized form; or
- a reference to another workflow script.

3.1.2 Workflow Interpreter

We have built an interpreter that takes a workflow script and produces a set of actions that implement the specified MP. This implementation involves activation of server and user agents of different types, the sequencing of their actions and the parameter loading and passing during those actions. The interpreter uses the following commmands:

- read workflow specification script,
- initialize variables,
- load defaults for variables and data, and
- execute workflow commands.

Initially, the workflow interpreter reads the main workflow script and starts executing the specified commands, controlling and sequencing the interaction between the intervening agents as well as loading and executing other possible workflow scripts specified in the main workflow.

Below we show two workflow script segments required to perform the outpatient care MP. In the first segment we can observe how the outpatient care MP is initiated by the *WFE*: once the doctor selects a patient, the workflow starts the interaction of the User agent with the Business Rule server agent for uploading the patient data into the user form (screen). Then the *WFE* calls for the interaction between the User agent and the Business Rule server agent for uploading relevant patient's EMR data, and the *WFE* initializes variables.

```
Procedure PreparePatient
  begin
    Interact(BRServerAgent(UploadPatientData));
    Interact(BRServerAgent(StartEMR));
    InitializeVariables();
  end
```

In a similar way, in the next segment, the *WFE* implements the *emit prescription* task (see figure 1) coordinating the interaction between specialized data base server agents, business rules server agents and user agents using specialized forms.

```
Procedure EmitPrescription
  begin
    Interact(BRServerAgent(DiagnosysCheck));
    DeactivateFields(StartField,EndField);
    Interact(BRServerAgent(SelectMedication));
    Interact(BRServerAgent(VerifyConstraints));
    Interact(BRServerAgent(GiveIndications));
    InputFields(AlergiesField,NextAppointmentField);
  end
```

3.2 The Outpatient Care Domain Layer

The outpatient care domain layer contains the following components:

Server Agents. These are software agents, owned by the organization, that are specialized in handling elements or components of the Hospital Information System. Currently, there are server agents for data bases, business rule repositories and workflow script repositories, but there may be other server agents when required by other applications or application domains. These agents act as front ends for all the repositories and devices of the hospital domain and thus handle the interactions with other domain agents.

User Agents These are software agents, also owned by the organization, that act as a front end for human users of the system such as patients, suppliers, hospital staff and eventually also for external processes.

Interaction Devices These are a type of device that we use to implement interfacing capabilities between user agents and their human counterparts (currently through forms, but eventually through other means) and in general between user agents and other server agents to perform activities like form handling, database calls and business rule triggering.

Workflow Scripts are a *workflow engine-interpretable* specification that define procedural behaviours of the organization.

Repositories include business rule repositories, workflow script repositories and other databases accessible to agents.

3.3 Executing the Outpatient Care Medical Protocol

As indicated in section 3.1.2, the protocol we have implemented is run by the middleware and workflow engine that interacts with the hospital domain components and human users. Figure 2 illustrates how the middleware and workflow engine reads workflow scripts from a repository and supervises the agents that handle the specialized domain components, such as databases or business rule definitions — a specialized *business rule* server agent (Bag) fetches, from a central repository, business rules that use data provided by another specialized *database* server agent (Dag), to provide input to a *user agent* (Uag) that displays it in a user form.

Once the interaction between the *user agent* and the *server agent* is established, the infrastructure makes sure that the communication between both agents is persistent until one of the agents decides to terminate it. The system is responsible for maintaining the context of all agent interactions because, as agent interactions evolve, they modify the context of the world, updating data and status variables as required.

4 Related Work

To define a system for monitoring MP, in the work described in [12], the researchers have made an *abstraction* of a hospital environment, in terms of a MAS. The main idea is to model medical services in hospitals as *Specialized Domain Agents* (SDA) and interactions between different services as *electronic communication processes*. From this point of view, a MP describes a negotiation process between multiple SDA for treating a particular pathology and specifies behaviour rules depending on specific symptoms.

The negotiation process is made possible by means of a formal specification language for modelling dialogical institutions, using a very similar concept to that described for electronic institutions in section 2.1. Due to the fact that *all* agent interactions are specified using a similar concept as EIs, the resulting definition of MP is cumbersome and limited for real world use. Our approach deals with this limitation using an organization engine and a grounding language that allows the system designer to handle different levels of abstraction in the specification of business rules, having a better trade-of between the complexity of business rules and the complexity of the workflow needed for agent interactions as explained in section 5. This issue is the main contribution of this paper, because we tested in a real application that it is possible – giving an adequate level of abstraction – to deploy an agent based application for representing and executing medical protocols. We can say similar things about the results we obtained in our previous work on hotel information systems as reported in [10].

5 Closing Remarks

In this paper, we have described the agentification of an outpatient care system. In the process, we have outlined the construction of the required server and user agents, developed the required business rules, and specified the workflow needed for the appropriate sequence of execution and a workflow engine that executes the workflow specification.

In our MAS-ified outpatient care protocol implementation, we have been able to separate the programming of business rules, user form definition and design, and the protocol definition via proper workflow specifications. We have found that with a system that allows this type of separtion it is easier to adapt its functional characteristics and implement the protocols involved.

In relation to the WF specification language explained in section 3.1.1, we have found that the specification and execution of medical protocols entails some balance between the complexity of the specified workflow and the complexity of the business rules (*BRs*) invoked in that workflow. Our experience with the outpatient care system tells us that if a protocol is specified using elaborate *BRs* that capture much of the discretional aspects of a a given process, the way that these *BRs* are put together is no more than simple sequencing. However, as *BRs* become simpler, the need for agent discretional behaviour is increased, the need for handling agent commitments emerges and interaction protocols become more involved. This suggests a trade-off between the complexity of the procedures implemented in the *BRs* and the complexity of the workflow specification needed to provide the proper sequence of execution between *BRs*, database repositories and user forms. If we implement elaborate procedures as *BRs*, then we need a very simple workflow definition, but, if we implement short and one-function procedures as *BRs*, we need a more sophisticated workflow definition. This is intuitively clear, because the more atomic the *BRs*, the more information is needed to control their interactions with other domain components.

We have also found that, if we do not have a mechanism to control the "rules of the game" and commitments generated by agent interactions, we need to hard-wire the required program logic inside each agent, and perhaps inside some *BRs*. This is because if agent a performs an action x at time t, it may be necessary that the same (or another) agent performs action y at time, say $t + 3$. If action x is implemented as a business rule, then we must have a mechanism to send a return value to the $BRagent$, or some way to set a variable in some kind of working memory.

Our current work is concerned with developing a framework that generalizes this example and profits from the lessons learned. More specifically, in order to deal with complex interactions, we aim at implementing prescriptive specifications that may be properly enacted. The driving intuition is the need for a high level specification of the intended operation of an organization that produces a machine executable counterpart that operates in an agentified business domain. Both the specification and the executable counterpart should be easy to deploy and update, hence the need for standard IT components in the business domain and a pervasive use of agents, and, likewise, the need for a rich and expressive environment for the specification of the prescriptive description and its run-time counterpart.

To achieve this, we are working with the notion of Electronic Institutions, and extending the current EIDE [1] environment to allow normative specification of agent interactions.

Figure 3 outlines our proposed extended framework. The diagram shows how our current middleware layer is to become an organization engine that implements the institutional conventions that will govern the business domain interactions.

We believe that the trade-offs in complexity that have become evident during the reported efforts, should be assessed against the flexibility we want our agentified systems to have. If hospitals or other organizations like them want to get a return on their investment in information systems, these systems need to be flexible enough to adapt to the dynamic conditions of their business environment with undue adaptation costs. This means that protocols and other knowledge rich system components need to be updated whenever the business environment changes. Hence the need for a methodology and tools that incorporate reusability of components and proper allocation of discretionality and organizational intelligence in different levels of the IT infrastructure. The framework we are working on moves in that direction and the outpatient care system described in this paper is a step forward.

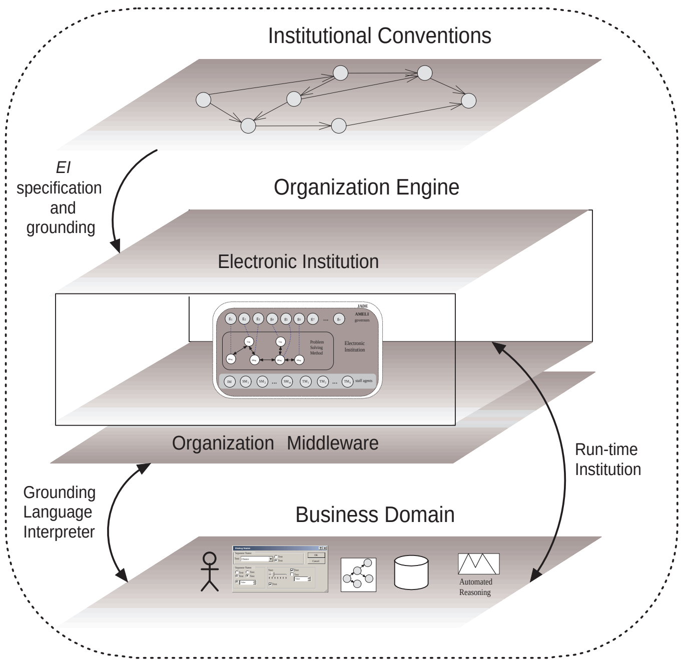

Figure 3. General Architecture using the concept of Electronic Institutions as part of the organization engine

ACKNOWLEDGEMENTS

This research is partially funded by the Spanish Ministry of Education and Science (MEC) through the Web-i-2 project (TIC-2003-08763-C02-00) and by private funds of the TCA Research Group.

REFERENCES

[1] Josep Arcos, Marc Esteva, Pablo Noriega, Juan Rodriguez-Aguilar, and Carles Sierra, 'Engineering open environments with electronic institutions', *Engineering Applications of Artificial Intelligence*, (18), 191–204, (2005).

[2] Richard M. Cyert and James G. March, *A behavioral theory of the firm*, Prentice-Hall, Englewood Cliffs, N.J. USA, 1963.

[3] Ada Valls David Snchez Ruenes, Antonio Moreno, *AgentCities.NET Deployment Grant #1*, Midterm report, Universitat Rovira i Virgili, Computer Science and Mathematics Department. Tarragona, Spain., September 27, 2002. $http://www.agentcities.org.$.

[4] Marc Esteva, *Electronic Institutions: from specification to development*, Ph.D. dissertation, Universitat Politècnica de Catalunya (UPC), Barcelona, Catalonia, Spain, 2003. Published by the Institut d'Investigació en Intelligència Artificial. Monografies de l'IIIA Vol. 19, 2003.

[5] James G. March and Herbert A. Simon, *Organizations*, John Wiley and sons, New York, USA., 1958.

[6] Pablo Noriega, *Agent Mediated Auctions: the Fishmarket Metaphor*, Ph.D. dissertation, Universitat Autònoma de Barcelona (UAB), Bellaterra, Catalonia, Spain, 1997. Published by the Institut d'Investigaci en Intelligncia Artificial. Monografies de l'IIIA Vol. 8, 1999.

[7] Douglass C. North, *Institutions, Institutional change and economic performance*, Cambridge Universisy press, 40 west 20th Street, New York, NY 10011-4211, USA, 1990.

[8] Lorenzo Boicocchi Silvana Quaglini Mario Stefanelli. Paolo Cicarese, Ezio Caffi, *A Guideline Management System*, Dipartimento di Informatica e Sistemestica, Universit di Pavia, Italy; Consorzio di Bioingegneria e Informatica Medica, Pavia ,Italy., 2004.

[9] A Robles and Pablo Noriega, ‘ A Framework for building EI–enabled Intelligent Organizations using MAS technology’, in *Proceedings of the Third European Conference in Multi Agent Systems (EUMAS05)*, eds., M.P. Gleizes, G. Kaminka, A. Nowé, S. Ossowski, K. Tuyls, and K. Verbeeck, pp. 344–354., Brussel, Belgium, (December 2005). Koninklijke Vlaamse Academie Van Belgie Voor Wetenschappen en Kunsten.

[10] A Robles, Pablo Noriega, Marco Robles, Hector Hernandez, Victor Soto, and Edgar Gutierrez, ‘ A Hotel Information System implementation using MAS technology’, in *Industry Track – Proceedings Fifth International Joint Conference on AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS (AAMAS 2006)*, pp. 1542–1548, Hakodate, Hokkaido, Japan, (May 2006).

[11] Juan A. Rodríguez-Aguilar, *On the Design and Construction of Agent-mediated Electronic Institutions*, Ph.D. dissertation, Universitat Autònoma de Barcelona (UAB), Bellaterra, Catalonia, Spain, 2000. Published by the Institut d’Investigació en Intelligéncia Artificial. Monografies de l’IIIA Vol. 14, 2003.

[12] R. Bejar C. Fernandez F. Manya. T. Alsinet, C. Ansotegui, *A Multi-Agent System Architecture for Monitoring Medical Protocols*, Department of Computer Science, University de Lleida, Jaume II 69, E-25001 Lleida, Spain., 1999.