www.itcon.org - Journal of Information Technology in Construction - ISSN 1874-4753

ORGANIZATION-ORIENTED MULTI-AGENT SYSTEMS FOR CONSTRUCTION SUPPLY CHAINS

REVISED: November 2011
PUBLISHED: November 2011 at http://www.itcon.org/2011/43
EDITOR: Turk Ž.

Esther A. Obonyo, Dr,
Rinker School of Building Construction, University of Florida, USA;
obonyo@ufl.edu

Chimay J. Anumba, Prof,
Architectural Engineering, The Pennsylvania State University, USA;
anumba@engr.psu.edu

SUMMARY: *The agent paradigm has been used by several researchers to build several complex construction industry-specific tools ranging from e-business applications to cross-disciplinary communication systems, supply-chain management and contract administration. Because of the slow progress in the maturity of the underlying infrastructure, such applications relied on platforms such as JADE based on agent-centered techniques which focus on the defining communication and interaction protocols based on individual agents' internal structure. An agent-centered approach undermines the potential for deploying a large-scale community of agents developed by different people. This paper discussed the potential for addressing this limitation using organizational abstraction to model multi-agent systems. It provides an overview of an agent-centered, JADE-based prototype system for construction specification and procurement. A critique of the agent-centered approach has been provided. The paper also reviews related work in organization-centered, agent systems. This is followed by a description of an organization-centered, JADE-based proof-of-concept based on construction products procurement requirements. The paper concludes with a discussion highlighting further work in the research. It specifically identifies the potential for enhancing the organization-centered MAS approach using ontologies.*

KEYWORDS: *Supply chain, e-business, intelligent agents, methodology, organizational abstraction*.

REFERENCE: *Esther A. Obonyo, Dr, Chimay J. Anumba, Prof (2011) Organization – oriented multi – agent systems for construction supply chains, Journal of Information Technology in Construction (ITcon), Vol. 16, pg. 727-744, http://www.itcon.org/2011/43*

INTRODUCTION

Despite the deployment of a number of advanced computing tools in several operations, the seamless flow of knowledge and information across the construction supply chain is still impeded by a number of factors. Data has to be drawn from dynamic, semi-structured and unstructured information and knowledge systems. A key issue with such systems is their open architecture (Miles et al, 1997, Sierhuis et al 2009, Habib and van Maurice, 2011). Their systems also exhibit dynamism - the components change and cannot be predetermined because of being implemented by different people, at different times, using different tools and different techniques. Interfacing with these repositories requires adaptive organizational models that can capture the highly dynamic, constantly changing business environment. The models must also capture interfaces with heterogeneous, distributed, applications. The heterogeneity in distributed computing applications has been further complicated by globalization – there has been an exponential increase in the number of projects being executed through synergies between international firms. Another significant factor is the ever increasing problem of information overload.

There has been an explosion in sheer volume of bytes available through advances in the generation real-time data using sensors. The information overload problem has also been compounded the additional digital information available through advances in data capture using of robotics and the emerging smart grid and other associated intelligent infrastructure systems.

In addition to altering organizational time scales, all these developments have also modified the construction supply chain through giving rise to new organizational forms, complexity and environment. (Preitula et al, 1998) Even though emerging sophisticated building information models have greatly enhanced knowledge sharing; they must still interface with several discipline-specific applications. Existing knowledge sharing tools have not adequately addressed the needs of such a complex business environment. Complexities inherent in generating explicit and independent representations of distributed applications' structures still remain largely unresolved (Sierhuis et al 2009). Addressing the outstanding issues calls for the development of dynamic organizational models that capture all the critical aspects of an open heterogeneous environment as summarized in Table 1 (Clancey, 1998; Clancey et al, 2002 & 2005; Klein et al, 2005 and Sierhuis et al, 2009)).

TABLE1: Modeling an Open, Heterogeneous Environment

Critical Aspects	Examples
Organizational structures	Dynamic entry and exit of actors
Behavioral complexities	Flexible roles, goals and tasks
Collaboration complexities	Formal and informal interactions, multiple teams or adhocracies through which actors perform individual or joint activities.
Regulatory components	Flexible representation of organizational norms, policies, laws and culture
Common understanding	Seamless flow of knowledge among actors
Context awareness	Using, reasoning and communicate about the physical and virtual environment

In the 1990s intelligent agent technology picked up momentum for its potential contribution to efforts directed at handling some of the complexities associated with using distributed applications. The definition of agent technology is not an easy task because the term 'agent' is a common term in every day conversion. It also encompasses a broad area. The various synonyms used by researchers further complicate matters (Nwana, 1996). Franklin and Graesser (1996) give a detailed analysis of several definitions by leading researchers, which will not be duplicated in this discussion. The working definition in this paper is: "intelligent agents are systems capable of autonomous, purposeful action in the real world" (Ferber, 1999). Agents present a distributed approach to locating, retrieving and integrating information. Their use therefore results in applications that can co-operate, co-ordinate and share their information with other applications. Being a relatively new field, the progress in this area has been slow but generally steady. A number of construction industry-specific, agent-based applications have been discussed by Anumba et al (2005). These illustrate different ways through which intelligent agents offer an innovative approach to overcoming knowledge and information sharing challenges in the construction sector.

Intelligent agents are usually used within the context of a multi-agent system (MAS), which is a computational system in which two or more (homogenous or heterogeneous) agents interact or work together to perform a set of tasks or to satisfy a set of goals (Lesser, 1999). Such a system comprises (1) an environment, (2) a set of passive objects that can be associated with a position in the environment, (3) an assembly of agents, which are specific objects representing active entities of the system, (4) an assembly of relations linking objects (and thus agents), (5) an assembly of operations with which agents perceive, produce, consume, transform and manipulate objects and (6) operators representing the assembly as well as reaction modifications (Ferber, 1999). A multi-agent system is therefore a consolidation of autonomous 'problem solvers'.

The notion of a community of agents cooperating to fulfill a complex task is the fundamental benefit of deploying agent-based systems. Interestingly, most of the existing applications are generally based on models which limit

interaction of agents across systems deployed by different developers. There has been a growing interest among researchers to resolve this problem through advancing modeling agent-based systems using organizational abstractions. In their research, Sierhuis et al (2009) identified specific examples of organizational-based approaches being used in MAS to address issues such as open organization modeling, argumentation frameworks, teamwork, and culture.

Agent-oriented organizational abstraction is still its infancy. There are no unified, robust organizational models that adequately capture organizational structures (with actors and roles), requirements and objectives. The emerging models can only be advanced to this status following their rigorous verification and validation in different use cases based on the requirements of specific disciplinary domains. This paper is directed at doing just that using the requirements of specifying and procuring construction products. It provides an overview of an agent-centered, JADE-based (URL1) prototype system for construction specification and procurement. A critique of the agent-centered approach has been provided. The paper also reviews related work in organization-centered, agent systems. This is followed by a description of an organization-centered, JADE-based proof-of-concept based on construction products procurement requirements. The paper concludes with a discussion highlighting further work in the research. It specifically identifies the potential for enhancing the organization-centered MAS approach using ontologies.

1. A REVIEW OF PIONEER AGENT-ORIENTED METHODOLOGIES

It is important to note that main stream research in the agent paradigm can be generally perceived as focused on either delivering the enabling infrastructures of implementing agent-based systems or deploying agent-based systems as context-specific applications. Obviously, the robustness of applications implemented by application developers largely depends on the maturity level of the enabling infrastructure. The systems discussed by Anumba et al (2005) constitute applications deployed using infrastructural platforms such as JADE (URL1). The motivation for using agents in these applications is the autonomy, flexibility and dynamism that agents can potentially add to the functioning of a knowledge and information system. However as discussed in the preceding section, their contribution to the successful the realization of truly open, heterogeneous systems has be largely limited by the lack of a framework for modeling and deploying the open multi-agent societies.

It is not easy to conceptualize advanced MAS concepts using conventional software engineering approaches. Consequently, a critical research issue for the agent community has been the definition of suitable methodologies for analyzing and designing a MAS. Since the 1990s, agent-oriented software engineering has attracted considerable of research interest resulting in a number of agent-oriented methodologies (Rana, 2001). This notwithstanding, existing agent-oriented methodologies are considered to be "academic" pursuits with the more advanced ones having gone no further than testing in small, industrial applications (Giorgini and Henderson-Sellers, 2005).

With the exception of a few rare deviations, many researchers in agent-oriented software engineering extende existing object oriented methodology to support agent concepts. Object abstraction introduced programming constructs for data encapsulation in modules, type extension through inheritance, and polymorphic substitution mechanisms for sub-types that supported improved structuring and organization of program modules over procedural techniques. In agent abstraction, extensions are made to standard, object oriented programming languages such as Java in order to facilitate advanced inter-agent communications mechanisms and directly implement agent management policies and semantics. Giorgini and Henderson-Sellers (2005) affirm that new programming language abstractions are the most concrete contribution of agent technology to software engineering. Examples of methodologies exploiting the object oriented paradigm include DeLoach's (2004) Multi-agent Systems Engineering Methodology (MaSE) and its derivative, O-MASE (DeLoach and Garcia-Ojeda, 2010), Gaia (Wooldridge et al, 2000 and Zambonelli et al, 2003], ADELFE (Bernon et al, 2003), the Methodology for Engineering Systems for Software Agents (Caire, 2001), INGENIAS (Pavón and Gómez-Sanz, 2003), Agent OPEN (Taveter and Wagner, 2005), Prometheus (Padgham, and Winikoff, 2002), PASSI (Cossentino et al, 2005) and the MAS-CommonKADS methodology (Iglesias et al, 1998 & 1999.

In a few cases, agent-oriented methodologies have been developed independently of object oriented

methodologies. Figure 1 shows Tropos as one such example. The Tropos software methodology exploits the "i* framework," which views organizations as collections of actors with strategic interests and interdependencies involving goals, tasks and resources (Yu and Mylopoulos, 1995).

With all these different agent-oriented software engineering methodologies being used to design interaction models, it is not surprising that the resulting applications amount to closed MAS as discussed in the first section. The main problem with the models is their focus on the internal mental state of individual agents. In the subsequent section the authors explain this problem further using models developed using the MaSE approach and deployed using JADE platform. This is followed by an assessment of the potential for advancing MaSE-like models using organizational abstraction.

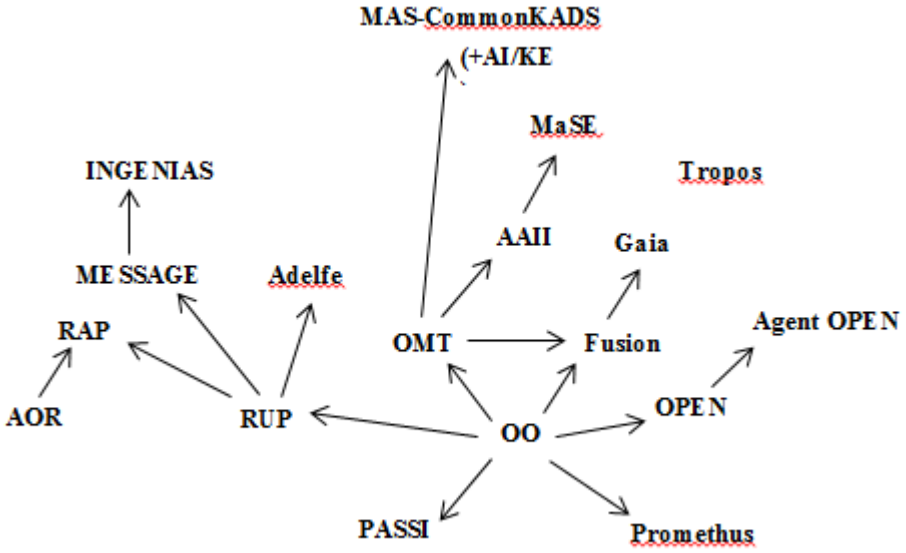

FIG 1: Influences of object-oriented methodologies on agent-oriented methodologies (Source: Giorgini and Henderson-Sellers, 2005)

2. MODELING MULTI-AGENT COMMUNITIES

3.1 Case Study of Agent-Centered Models: Conceptualizing Agents for Construction e-business using MaSE

An agent-based application was deployed based on the requirements for construction specification and procurement - see Obonyo (2005) for a detailed discussion. The simulated use case was processing product information for the specification and procurement of light bulbs from the Philips Lighting Web site (URL2). The site hosts close to 200 catalogues in Adobe Acrobat PDF format. The information that would be of interest to an end-user such as wattage, cap size and voltage is presented in a semi-structured format. The Web site does not have a search facility supporting guided navigation based on attributes such as wattage and voltage. It is also not possible to query the information directly from any another application. Furthermore, relevant data has to be re-keyed for reuse in a different application.

It was necessary to review the existing agent deployment infrastructure as it affected the scope of the defined problem. Some of the existing agent development environments were used to develop the early versions of the proposed prototype. This exercise revealed that no single agent development environment was sufficiently developed to support the deployment of an entire system with the desired functionality. Consequently, the envisaged system was re-defined into components for implementation as linked modules. Developers of agent-based applications leverage on existing infrastructure. The selection of the tools was influenced by the need

to develop an ensemble of java-based components. The work presented in this section is based on the design of a MAS the MaSE methodology that has been depicted in Figure 1 as having objected oriented roots. The selection of the MaSE methodology for the design and development of an exemplary MAS application for Construction e-business was largely influenced by the fact that its use is supported by user-friendly, Java-based deployment application (agentTool). The authors were confident that this approach would result in models that can be adapted for use with other Java-based development platforms such as JADE, which emerged as the most robust platform the selected use case.

As depicted in Figure 1, the MaSE approach is abstraction derivative of the tradition object oriented paradigm. In MaSE models, agents are a specialization of objects (DeLoach, 2001; DeLoach and Matson, 2004). Unlike simple objects whose methods can be evoked by other objects, agents in this framework coordinate with each other via conversations and act proactively to accomplish individual and system-wide goals. The MaSE models define the different types of agents in the community, their interfaces to each other and the internal agent design. The resulting models for different levels of functionality were translated into visual representations using the agentTool (URL 3). These models and visual representations have been outlined in the subsequent paragraphs.

Agent Goals: The functionality for the envisaged use case is achieved through the activities of agents in the system. Figure 2 depicts a hierarchical view of goals to be achieved by the system. Clearly, some requirements are presumed within other agent roles. For example, though not explicitly shown in the model, a mechanism for triggering the system to start up to interface with the user is presumed.

Agent Roles and Tasks: The formal specification of the required structure and behaviour for the envisaged multi-agent community also involved the definition of high-level system behaviour. This definition identified the different types of agents in the system and the communications between them. This resulting system-level specification was then refined for each type of agent in the system. In order to execute the functions outlined in Agent Goals model, different categories of agent roles were created. Sequence diagrams such as the one shown in Figure 3 were used to model use cases for the construction e-business domain. Search Agent and a Resource Broker lie at the core of the system in the depicted exemplary use case. When a user inputs search parameters, the system seeks a list of available Search Agent for the specified functional area from the Resource Broker. The system then forwards the user's request to the available Search Agents and waits for response from the individual searches. Upon receiving the requests, the Search Agent notifies the Resource Broker of its increased workload. After each Search Agent has completed the search, the results are ordered and reported to the user. The Search Agent notifies the Resource Broker of the job completion.

The identified agent roles were extended to allow the definition of each agent's tasks and protocols as shown in Figure 4. In the depicted model, each agent role has direct references to previously identified goals for the system. For example, by embedding the "1.1.2.1 label" within the role of a Resource Broker, the model identifies this agent as holding the primary responsibility for determining available systems for searching.

Agent Classes: A major step in the MaSE Approach involves capturing the defined agent roles in agent classes and defining the communication protocol between the different classes. This is accomplished by creating Agent Classes in which the internal workings of each agent can be defined as shown in Figure 5. The agent class 'Broker' (assuming the role of a Resource Broker) shown in diagram works in collaboration with two other agent classes, namely 'SearchInterface' (assuming the role of the System) and 'ProductFinder' (assuming the role of a SearchAgent).

Agent Communication: Unlike simple objects whose methods can be evoked by other objects, agents in the defined framework coordinate with each other via conversations and act proactively to accomplish individual and system-wide goals. The model in Figure 5 shows how MaSE was used to define how the different agent categories communicate with one another.

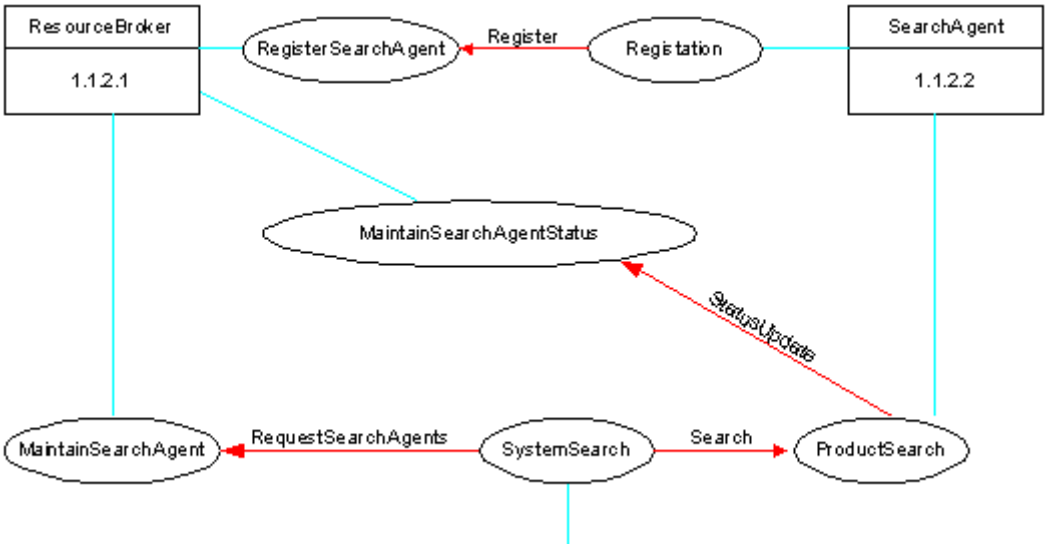

FIG 4: Agent Roles

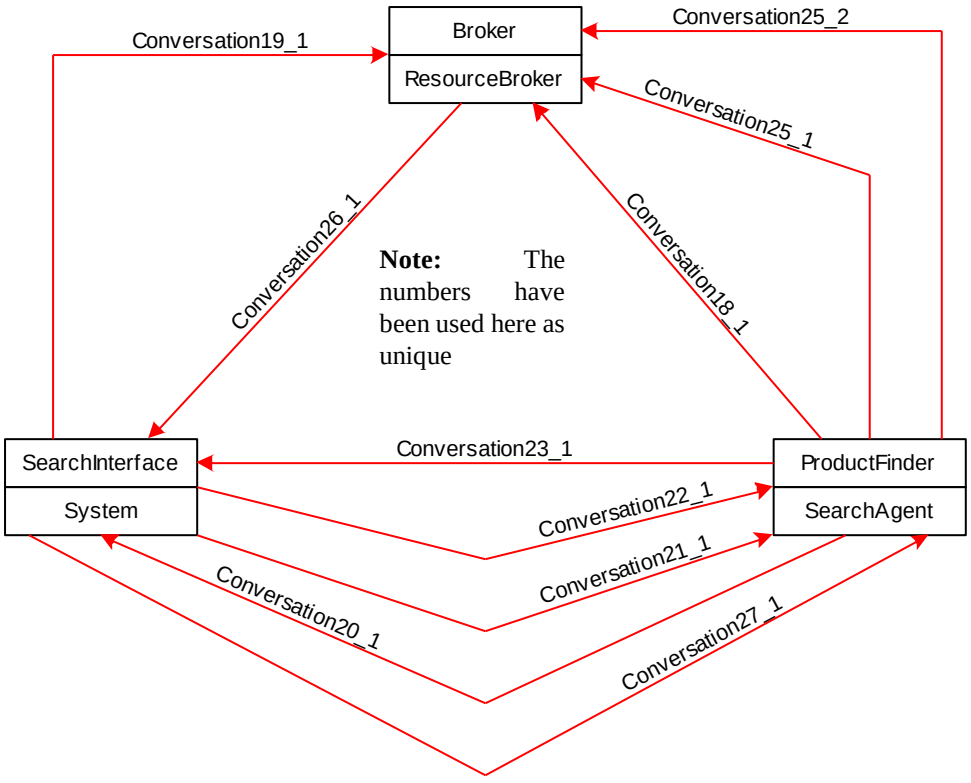

FIG 5: Inter-Agent Communication

The depicted agent conversations are based on the previously defined communication protocols between the different agent classes and roles as shown in Table 2. Clearly, the MaSE methodology allowed the definition of high level system behaviour captured as agent goals to be extended into low level tasks to be executed by individual agents.

TABLE 2: Agent Communication

Agent	Role	Task	Conversation
Broker	ResourceBroker	Register Search Agent	18_1
		Manage Search Agent	19_1 & 26_1
		Maintain Search Agent Status	25_1
ProductFinder	SearchAgent	Registration	18_1
		Product Search	20_1, 21_1, 22_1, 23_1, 25_1, 25_2 & 27_1.

The resulting set of requirements from the MaSE methodology was synthesized and mapped into the conceptual architecture depicted in Figure 6.

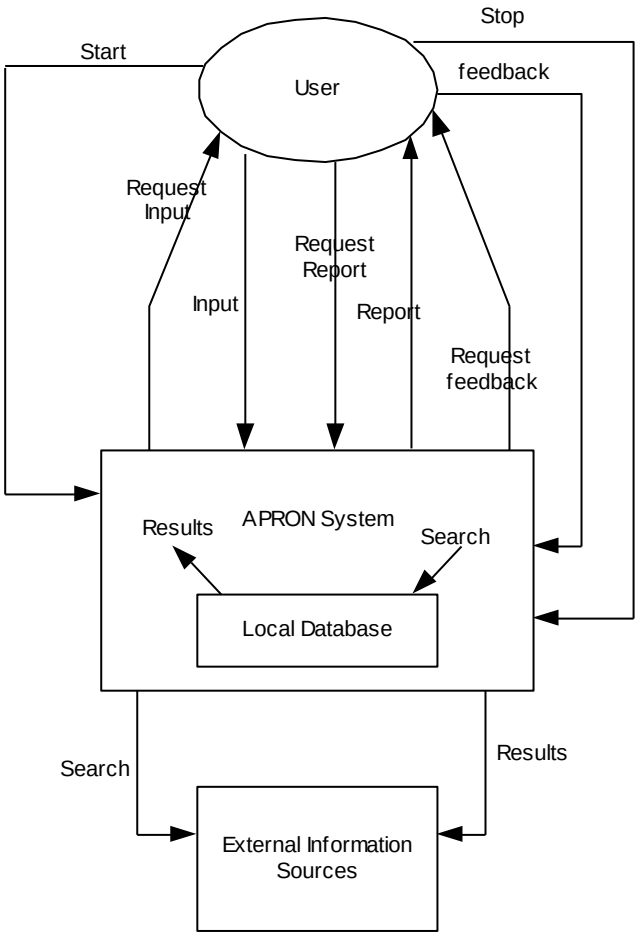

FIG 6: Functional Components

This architecture was used as a launching pad for initializing the prototyping phase. The required system-level behaviour for the resulting multi-agent system was a sum total of coordinated functions of linked modules. A detailed description of the implementation and testing of the proof-of-concept has been provided in (Obonyo et al 2004, 2004, 2005a & 2005b). Based on the foregoing discussion it is clear that communication within the resulting MAS is driven by the internal mental state of individual agents.

3.2 A Critique of MaSE-Like Models

The agent paradigm allows a system to be decomposed into modules of self-interested agents which cooperate to execute complex tasks such as the ones inherent in construction e-business. Although the Construction e-business MAS architecture described in the preceding section was largely based on this notion, the resulting application was, in some respects, a closed system in the sense that the deployed agents would not easily interact and cooperate with agents implemented by different developers. The MaSE-based modelling approach discussed in the preceding section constitutes an agent-centered approach, and therefore results in 'agent-centered multi-agent systems' (ACMAS). ACMAS-driven research focuses on the internal mental state of an agent, the relationship between these states and its overall behavior. In ACMAS communications become speech acts whose meaning can be described in terms of the mental states of an agent – this is evident in agent communication languages such as the KQML and FIPA ACL (Ferber et al, 2003). The MaSE models in the preceding section depict concrete examples of such mental states. This is in effect what limits agents in applications deployed using this approach to communication with one another within a closed system (Ferber et al, 2003 and Zambonelli et al, 2001).

The main benefits of defining a societal structure for MAS include reducing the system's complexity, increasing the system's efficiency, and enabling more accurate modeling of the problem being tackled (Jennings and Wooldridge, 2000). Having agents that can only interact within a closed MAS complicates the definition of such a societal structure. Without this societal structure patterns and outcomes of agent interactions become inherently unpredictable - predicting the behavior of the overall system based on its constituent components is extremely difficult (sometimes impossible) because of the high likelihood of emergent (and unwanted) behavior (Jennings, 2000). In such a scenario the interaction ofagents designed by different designers will require making some assumptions about the primitives of communications (the "performatives" of the language) and the architecture of agents have to be made (Ferber et al, 2003). For example, one can assume that agents behave purposively in a cognitive way, using some kind of BDI (Belief-Desire-Intention) architecture. Agents based on ACMAS models do not have access to these constraints, which are often specified as ISO-like standards. The agents cannot therefore accept or refuse to follow these standards. These agents are therefore constrained to communicating using the same language and have to be built using very similar architectures. Moreover, such an approach lack generality and is tuned to specific systems and agent architectures - it exploits abstractions that are unsuitable for modeling agent-based systems (Zambonelli et al 2003; Iglesias et al, 1999; Bussman, 1998; Ferber and Gutknecht, 1998, Ferber, 2005)

There are three other major weaknesses of using MaSE-like, ACMAS-based models (Ferber et al 2003). Firstly, since all agent communications are without any external control, an applications designer often has to balance between: 1) allowing free interaction of agents thus making it easy for an agent to act as a pirate and use the system fraudulently, and; 2) implementing too strong security measures that could prevent the system from working efficiently in domains where speed and response is of critical concern. Secondly, with ACMAS all agents are accessible from everywhere, grouping entities that work closely together into "packages" that may or may not be hidden as is done in software engineering. The actual challenge within a MAS model is coming up with a dynamic framework for grouping agents that need work together. Finally, since in the ACMAS model the platform is the only supported framework, it is not possible to exploit the "component concept," something that is used in classical engineering as an abstract architecture in which components plug-in and its use often coupled with a definition sub-frameworks. All these factors limit interaction of agents with other agents outside their MAS.

ORGANIZATIONAL ABSTRACTIONS IN CONSTRUCTION E-BUSINESS

4.1 Contextual Background

Leading researchers in agent-oriented methodology have tried to address the limitations of ACMAS models using macro-level concepts such as 'organizations', 'groups', 'communities', 'roles' in designing multi-agent systems (Ferber and Gutknetcht, 1998; Omicini, 2001; Parunak and Odell, 2002; Zambonelli and Parunak, 2002; Coutinho et al, 2010 and Dinu et al, 2010). An organization is viewed in this context as structured aggregates of several, potentially decomposable and possibly overlapping partitions (Ferber, 2003). An agent-based system such as the one designed by the authors for construction specification and procurement (Obonyo et al, 2005a) can actually be

viewed as several interacting organizations and it is in fact possible for an agent to be part of multiple organizations (Zambonelli et al, 2003). It is therefore not surprising that organizational constructs are being perceived as being the first-class entities in MAS giving agent-based systems computational mechanisms for flexibly forming, maintaining and disbanding organizations. This implies that the notion of a primitive component can be varied according to the needs of the observer. Additionally, such structures provide a variety of stable intermediate forms in which individual agents or organizational groupings can be developed in relative isolation and then added into the system in an incremental manner (Jennings, 2000).

Pioneer work in this area dates back to conceptual ideas by Jennings (1999). Zambonelli et al (2003) affirmed this view, specifically stating that since agent-based systems can be naturally viewed as computational organizations, organizational abstractions and the associated metaphors and concepts should play a key role in the analysis and design of MAS. Organizational-centered multi-agent systems (OCMAS) can build on the mental states of ACMAS. However, since these mental states limit the definition of a societal structure for MAS, such mental states can be avoided altogether. Ferber et al (2003) demonstrated that it is possible to design MAS using only organizational concepts such as roles (or function, or position), groups (or communities), tasks (or activities) and interaction protocols (or dialogue structure). This in effect gives developers the ability to build organizations as frameworks where agents with different cognitive abilities may interact. Based on such a model, the resulting MAS application can then truly reflect the dynamic and flexible characteristics of an open system. Figure 7 shows the typical organizational characterization of MAS. The essential concepts of the model are: agents, high level interactions and organizational relationships.

Since the behavior of agents within a system is based on the behavior and structure of human organizations, each agent has a clearly defined role within the system. Consequently, interactions are no longer mere expressions of classical object oriented, interdependencies (Booch, 1994), but are also a characterization of the position occupied by an agent within the organization. The organizational metaphor also simplifies the design of the system by separating the component level (intra-agent) design dimensions from the system level (inter-agent). An additional benefit of the organizational metaphor is the ease with which the designed MAS can be closely mirrored to real-world organizations they are intended to support (Ferber, 2003).

The initial phases in the organizational abstraction process include (Ferber, 2003): 1) analyzing how the organization is supposed to work, 2) designing an organization which best fits the requirements, and 3) determining whether or not there is a potential for re-using available components. These concepts are captured in organizational abstraction as “organization rules,” “organization structure” and “organization patterns.” Organizational rules define the general and global (supra-role) constraints required for the proper instantiation and execution of MAS (Zambonelli et al, 2003; Zambonelli and Parunak, 2002). These rules also describehow the organization is expected to work. Within the context of open systems, organizational rules are used to manage the arrival of new, previously unknown and possibly, self-interested agents through explicitly defining: whether and when to allow newly arrived agents to enter the organization and once accepted, what their position in the organization should be; which behaviors should be considered as an expression of self-interest, and which among them should be prevented (Zambonelli et al, 2003; Zambonelli and Parunak, 2002). The “organization structure” defines the specific class of organization and control regime to which agents or agent roles have to conform in order for the entire MAS to work efficiently and according to its specified requirements. Pre-defined and widely used organizational structures that can be re-used from system to system are defined and expressed in organization patterns.

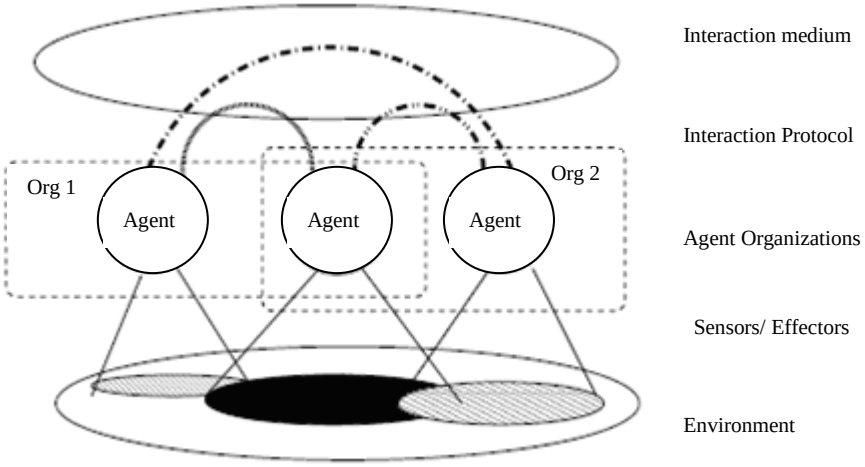

FIG 7: Characterization of Multi-Agent systems
(Source: Jennings, 1999)

An example of agent interaction through organizational abstraction is the Agent/Group/Role (AGR) model (Ferber et al, 2003; Ferber and Gutknetcht, 1998). In this model an agent is specified as an active, communicating entity, which plays (multiple) roles within (several) groups while organisational concepts such as groups, roles, structures and dependencies, are first-class citizens of a design methodology (see Figure 8).

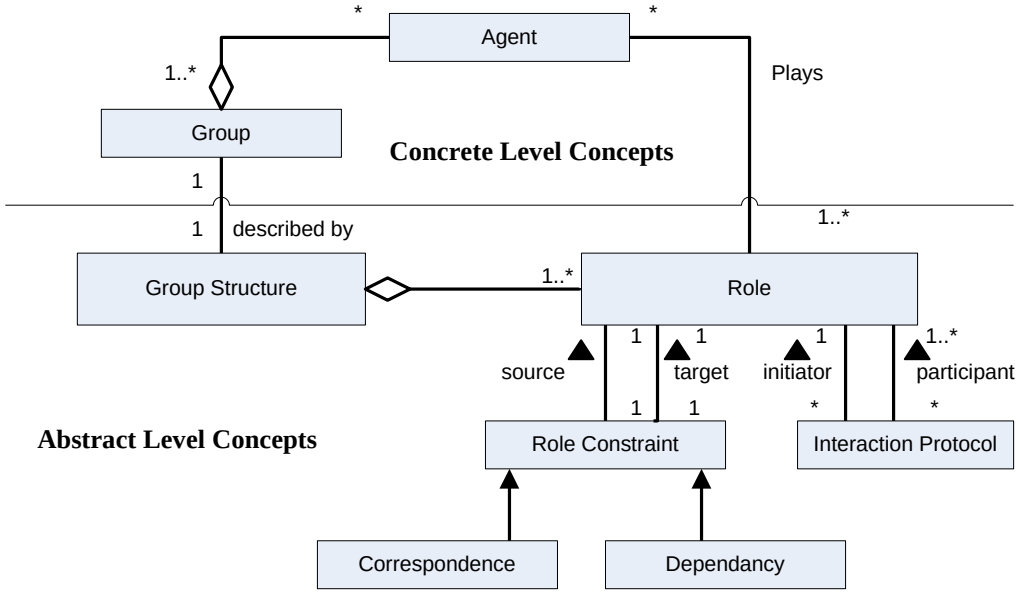

FIG 8: The AGR model
(Source: Ferber et al, 2004)

No constraints are placed on either the architecture of an agent or its mental capabilities. This model is divided into concrete level concepts (agent, role, and group) and abstract level concepts (possible roles, interactions and organizational structures). Agents can join and leave as many organisations as they need to, playing different roles in the different groups since each organisation is defined solely using the arrangement of groups and roles rather than using any individual agents' actual behaviour. Different types of requirement patterns and the associated logical relationships can be defined and formalised. The definition encompasses single role behaviour requirements, intergroup role interaction requirements, intragroup communication successfulness requirements, and intragroup role interaction requirements).

4.2 Case Study: OCMAS Models for Agent-based Construction e-business

In a preceding section, the authors described construction e-business MAS models in which specification and procurement sub-tasks have been delegated to a community of agents. Each agent within the system had clearly defined roles thus controlling a given phase of the e-business process. The primary goal of the entire system is achieved through its constituent agents communicating using predefined interaction protocols. The role of each agent is ensuring that a given phase of the e-business process works as intended. Each agent thus needs to sense and effect the subset of the environment representing the phase they are responsible for. In order to contribute to the global goal of enabling the specification and procurement of construction products, the constituent agents must interact with one another. In the subsequent paragraphs, organizational concepts at a macro level are used to specify and model an organization structure and the global constraints for construction e-business using the AGR model. The envisaged OCMAS architecture addresses the need to model open and global constraints.

A key attribute of Ferber's (2003) AGR model is its minimalist, structure-based view of organizations - a role-group structure is imposed on agents. Although AGR allows agents to have their joint behavior orchestrated by interaction protocols, it leaves open the nature and the primitives used to describe such protocols. Figure 9 depicts a MAS organizational structure for Construction e-business with interactions between three group structures: a procurer group structure (ProcurerGS); the manufacturers group structure (ProviderGS), and a contracts group structure (ContractGS) used when a client decides to buy a product from the provider.

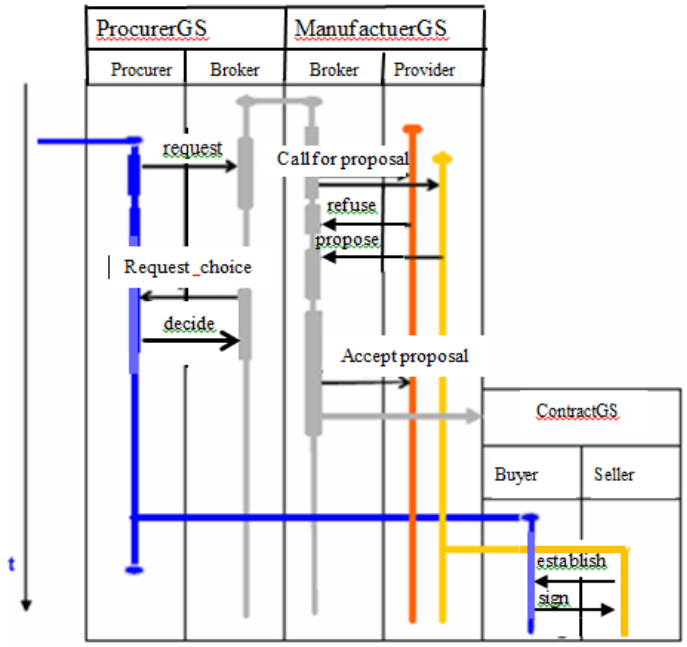

FIG 9: Construction e-business example
Adapted from Ferber's (2003) AGR Model

The agent role labeled "broker" for two group structures is actually the same agent with dual roles in the client group and the manufacturer group. The interaction is executed as follows:

- An agent enters a client group,
- The client asks the broker for a product.
- The broker sends a call for proposal to manufacturers.
- The resulting proposals are presented to the client for product selection.

- If a suitable match, a contract group is created with both the client and the chosen manufacturer, taking the respective role of “Buyer” and “Seller.”

The definition of organization-oriented models was advanced to a proof-of-concept for an exemplary scenario using MadKit (URL 4). MadKit is an open source application that supports the deployment of agents based on organizational-centered models. The proof-of-concept was based on information requirements for accepting bids from and selecting a concrete subcontractor. Autonomous agents were used to represent real-world parties and play the following roles: contractor (in this case concrete subcontractor, material broker and material supplier (offering precast and cast in place options). In a MadKit-like, organization centered approach, the interaction of agents is managed using a framework that encompasses all the roles to be executed by multiple agents from different sources representing multiple contractors, material brokers and material suppliers. Exchange of information is brokered through linking each agent to a specific role in a specific role in a given community (Figure 10). Different types of agents developed using any Agent platform can be modeled as distributed applications in MadKit through defining the existing communities or groups (for example, Construction-MaterialProvider and Construction-Contractor), the roles within this group and then link the existing agents to these roles. The same agent can play different roles in different communities. Agents belonging to different communities can also interact with one another.

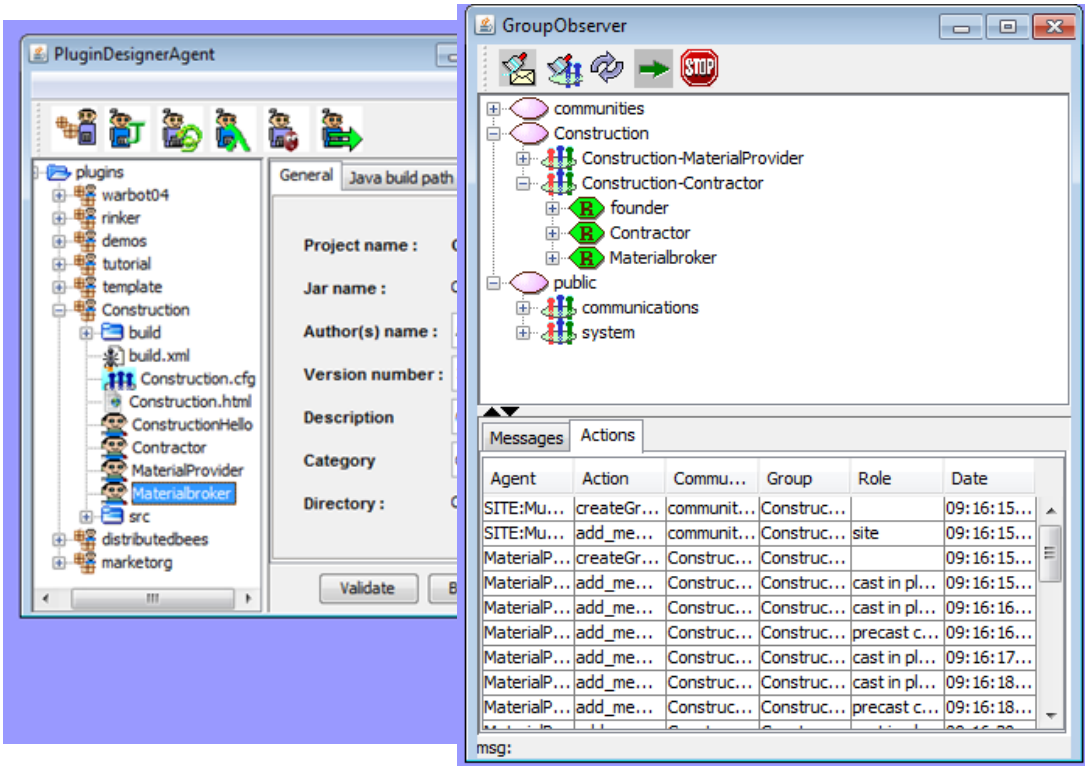

FIG 10: Agent Roles and Groups in Concrete Procurement

For the proof-of-concept, interaction is initiated by the agent representing the contractor sending out request for bids for both precast elements and cast in place concrete. When the Contractor agent is launched it interacts with the Materialbroker agent based on the definition of roles. The Materialbroker agent invites bids from agents assigned the role of Supplier agent. For this use case, supplier agents were placed in a different community (Construction-MaterialProvider) to demonstrate communication across groups. The existing Supplier agents offer material deals to the Materialbroker agent who returns the lowest bid to the Contractor. The selection criteria can either be coded in advance or provided during the time of the interaction (Figure 11).

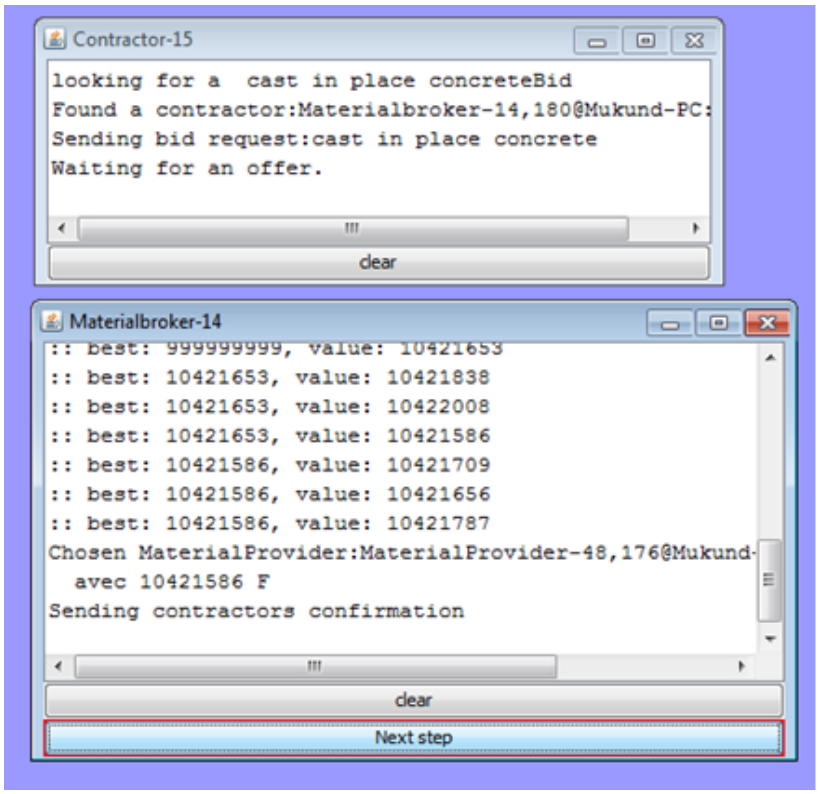

FIG 11: Agent Interactions Concrete Procurement

3. DISCUSSION AND CONCLUSIONS

Knowledge and information systems need to be designed for the deployment of open MAS to fully exploit the flexibility and autonomy of software agents. A key barrier to the realization of open MAS is the use of agent-centered models. The authors have explained this limitation using examples of agent-centered, MaSE models and JADE agents. These models and agents were developed by the authors in 2002 using was, at that time, cutting edge as far as enabling infrastructure is concerned. An important accomplishment at that point was the use of the MaSE approach resulted in the conceptualization of a multi-agent community as an ensemble of linked components for implementation in different agent development environments. JADE provided the unifying platform for all the agents. Although the authors were successful in developing, wrappers within JADE to create a unifying platform for agents implemented using different agent development environment, the use of agent-centered, MaSE-like models when coupled with implementation in an environment such as JADE yields a closed MAS. The main problem applications designed and implemented using such an approach is the definition of agent interactions. For example, within JADE, shared understanding is achieved through having a directory agent coordinating the interaction and packaging messages into the FIPA compliant ACL communication protocol. There is therefore a large emphasis on the internal structure of each individual agent during the design process. The negative impact of this is that the agents in the resulting multi-agent community cannot communicate easily with agents developed outside of JADE. This creates a bottleneck as far as seamless flow of knowledge and information is concerned. In an open heterogeneous environment, there will be distributed applications with components (including agents) deployed using different development environments.

In subsequent research work, the authors have been assessing the potential for deploying large-scale, real-life, agent-based applications within the context of construction e-business through identifying strategies that would enable agents in a given MAS to autonomously interact with agents developed by other people thus reflecting truly open systems. Organizational abstraction has been identified as a potential solution. It is based on the use of

organizational metaphor to encapsulate macro concepts in the design of agent-based systems thus making the deployed applications open and dynamic systems.

In the preceding sections, examples of organization-centered, AGR models and MadKit agents have been provided. These examples of models and an OCMAS-based proof-of-concept illustrate specific ways through which the organizational metaphor can contribute to the definition of a truly open MAS. Being an emerging area, it is still not possible to develop robust organizational models. Additional research work is required to reduce the gap between the real world and the design models used in agent-oriented software engineering. From a recent review, the authors established that leading researchers in this area are continuing to make significant advances towards the realization of stable infrastructure. There are also some semantic barriers to be addressed on the application development side. Before the organizational metaphor can take root as a superior approach to analyzing and designing agent-based system, semantic complexities inherent in the use of the OCMAS model will have to be addressed. As organizational abstraction becomes truly global and models interaction between agents in different groups implemented by different developers, there will be a need for reconciling differences in the use of concepts and terms. A possible solution is defining an ontology for MAS organizational concepts. Figure 12 shows an example that can be easily adapted to fit the requirements of construction industry-specific e-procurement.

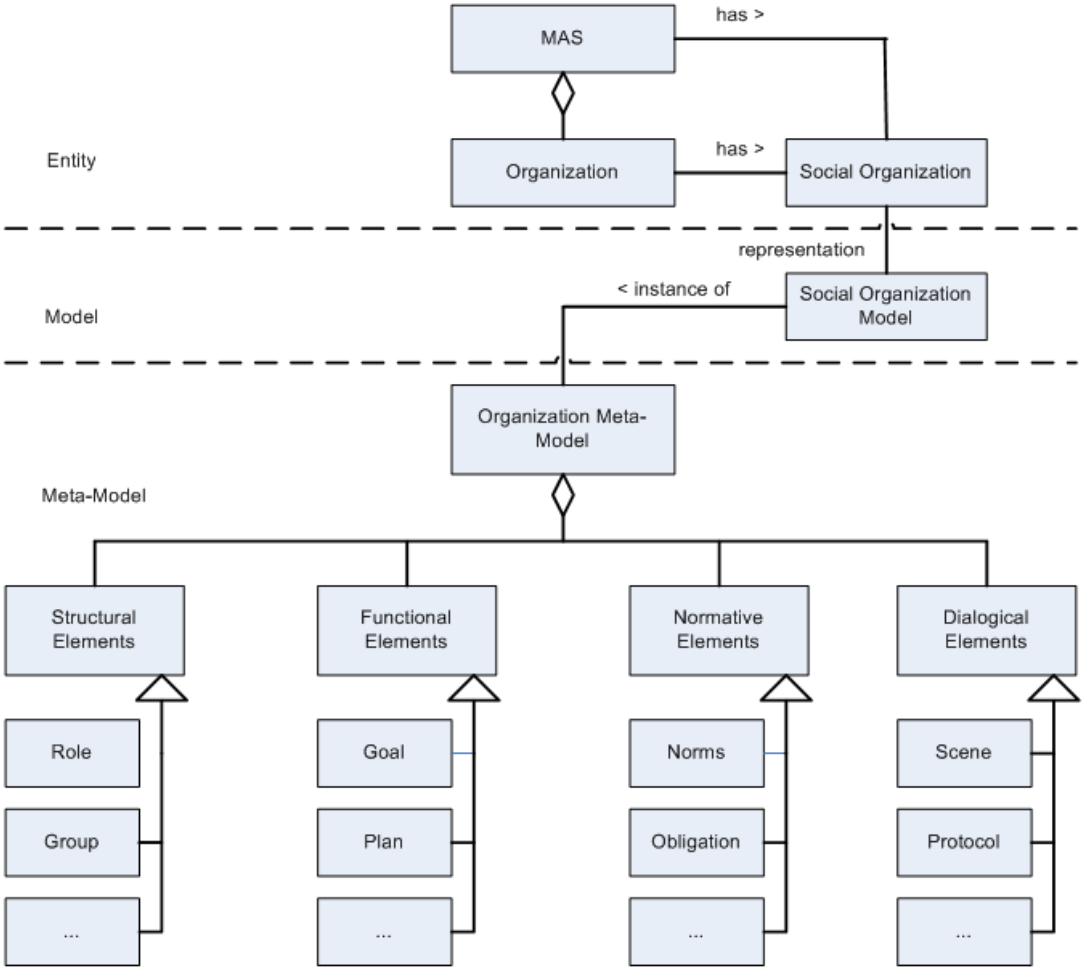

FIG 12: An Ontology for Organizational Concepts
(Source: Coutinho et al, 2005)

The framework is based on Coutinho et al's (2005) conceptual ontology, which has an entity level, a model level and a meta-model level. The entity level has representations of the "Organization" concept (an entity with identity that is materialized by a collection of agents) and the Social Organization concept (constraints that shape the actions and interactions of a collection of agents). The Social Organization model conforms to an Organizational Meta-Model with 4 main categories of modeling elements as shown in the diagram. This conceptual ontology can be extended into a more detailed ontology for using the OCMAS model in construction industry-specific e-business.

From the foregoing, it is clear that organization abstraction can be used to advance the seamless flow of knowledge and information across dynamic, heterogeneous, distributed applications. This notwithstanding, irony is not lost here: the successful deployment of an open MAS in which agents deployed by different developers can interact will inadvertently introduce additional semantics problems. The use of open and dynamic models based on the organizational metaphor to deploy an open MAS within a networked supply chain implies that information agents roaming on the Web can join and leave groups designed by other developers using different terms. The successful use of organizational abstractions in large scale knowledge and information systems will therefore require the development of some mechanism for managing differences the use of concepts. Follow-up work in this research, the authors will assess the potential for addressing this semantic problem through defining an ontology for an organization-centered, construction e-business, agent-based application.

4. REFERENCES

Anumba C. J., Ugwu O. O. & Ren Z (2005), *Agents and Multi-Agent Systems in Construction*, Taylor & Francis Books Ltd, London.

Bernon, C. Gleizes, M. Peyruqueou, S. and Picard, G. (2003), ADELFE: A Methodology for Adaptive Multi-agent Systems Engineering, *Engineering Societies in the Agents World III: Third International Workshop, ESAW 2002*, Madrid, Spain, September 16-17, 2002. Revised Papers, Springer Berlin / Heidelberg, Vol. 2577/2003, pp156-169.

Booch, G. (1994), Object-oriented analysis and design (second edition), Addison Wesley, Reading, MA.Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F. and Mylopoulos, J. (2001), *Modeling Early Requirements in Tropos; a Transformation Based Approach*, AOSE 2001, pp 151-168.

Bussmann, S. (1998), Agent-Oriented Programming of Manufacturing Control Tasks, *In Proceedings of Third International Conference on Multi-Agent Systems (ICMAS'98)*, IEEE Computer Society, pp 57-63.

Caire, G. Chainho, P. Evans, R. Garijo, F. Gomez Sanz, J. Kearney, P. Leal, F. Massonet, P. Pavon, J. and Stark, J. (2001), Agent Oriented Analysis using MESSAGE/UML, In *Agent-Oriented Software Engineering II*, Springer Verlag, 2nd International Workshop (AOSE).

Clancey, W., Sierhuis, M., Damer, B. and Brodsky, B. (2005), The cognitive modeling of day in the life social behaviors using brahms, in *Cognition and Multi-Agent Interaction*, R. Sun, Ed. New York, NY: Cambridge University Press, pp 151–184.

Clancey, W.J. (2002), Simulating activities: Relating motives, deliberation, and attentive coordination, *Cognitive Systems Research*, Vol. 3(3), pp 471–499.

Coutinho, L.R. Jaime, S. Sichman, J.S. and Boissier, O. (2005), Modeling Organization in *MAS: A Comparison of Models, First Workshop on Software Engineering for Agent-oriented Systems*, SEAS.

Cossentino, M., Gaglio, S., Sabatucci, L. and Seidita, V. (2005), The PASSI and Agile PASSI MAS Meta-models Compared with a Unifying Proposal. *CEEMAS 2005*, pp183-192.

DeLoach, S. A., Wood, M. F and Sparkman, C. H. (2001), Multiagent Systems Engineering, *The International Journal of Software Engineering and Knowledge Engineering*, Vol. 11(3), pp. 231-258.

DeLoach, S.A. and Matson, E. (2004), An Organizational Model for Designing Adaptive Multiagent Systems, *The AAAI-04 Workshop on Agent Organizations: Theory and Practice (AOTP 2004)*, Technical Report WS-04-02, AAAI Press, pp. 66-73.

DeLoach, S.A. and Garcia-Ojeda, J.C. (2010), O-MaSE: a customizable approach to designing and building complex, adaptive multiagent systems. *International Journal of Agent-Oriented Software Engineering*. Volume 4, no. 3, 2010, pp. 244 - 280.

Debenham, J. and Henderson-Sellers, B. (2002), Full lifecycle methodologies for agent-oriented systems the extended OPEN process framework. In *Proceedings of Agent-Oriented Information Systems (AOIS-2002)*.

DeLoach, S. A. (2004), The MaSE Methodology, In F. Bergenti, M. Gleizes and F. Zambonelli (eds.), Methodologies and Software Engineering for Agent Systems, *The Agent-Oriented Software Engineering Handbook Series: Multiagent Systems, Artificial Societies, and Simulated Organizations*, Vol. 11. Kluwer Academic Publishing.

Dinu, R., Stratulat, T. and Ferber, J. (2010), A formal approach to MASQ, In *Proceedings of AAMAS'2010*, pp.1443~1444

Ferber, J. and Gutknecht, O. (1998), Aalaadin: A meta-model for the analysis and design of organizations in multi-agent systems, in *Third International Conference on Multi-Agent Systems, IEEE*, pp128-135.

Ferber, J. Gutknecht, O. and Michel, F. (2004), From Agents to Organizations: an Organizational View of Multi-Agent Systems, in *Agent-Oriented Software Engineering (AOSE) IV*, P. Giorgini, J. Müller, and J. Odell (eds), Melbourne, July 2003, LNCS 2935, pp. 214-230.

Ferber, J., Gutknecht, O., Michel, F. (2003), From Agents to Organizations: An Organizational View of Multi-agent Systems. *AOSE 2003*: pp 214-230.

Ferber, J. (1999) Multi-Agent Systems, *An Introduction to Distributed Artificial Intelligence*, Addison-Wesley, An imprint of PEARSON EDUCATION, UK.

Franklin. S. and A. Graesser, A. (1996), Is it an Agent, or just a Program? A Taxonomy for Autonomous Agents, In *Proceedings of the Third International Workshop on Agent Theories, Architectures and Languages*, Springer-Verlag.

Giorgini, P. and Henderson-Sellers, B. (2005), Agent-Oriented Methodologies: An Introduction, In P. Giorgini and B. Henderson-Sellers (eds), *Agent-Oriented Methodologies*, Idea Group Publishing, London, UK.

Habib, M.B. and van Maurice, K. (2011), *Information Extraction, Data Integration, and Uncertain Data Management: The State of The Art.* (Internal Report)

Iglesias, C.A. Garijo, M. and Gonzalez, J.C. (1998), A Survey of Agent-Oriented Methodologies in Intelligent Agents V, Proceedings of the Fifth International Workshop on Agent Theories, Architectures, and Languages (ATAL-98), *Lecture Notes in Artificial Intelligence*, Vol. 1555, Springer-Verlag, Berlin Heidelberg New York.

Iglesias, C., A., Garijo, M. and Gonzales, J. (1999), A survey of agent-oriented methodologies, in Intelligents Agents IV: Agent Theories, Architectures, and Languages, *Lecture Notes in Artificial Intelligence*, Vol. 1555, Springer-Verlag, pp. 317-330.

Klein, G., Feltovich, P., Bradshaw, J. and Woods, D.D. (2005), "Common ground and coordination in joint activity," in Organizational simulation, W. Rouse and K. Boff, Eds. New York: Wiley-Interscience, pp 139–184.

Jennings, N.R. and Wooldridge, M. (2000), Agent-Oriented Software Engineering, In Bradshaw, J. ed. *Handbook of Agent Technology*, AAAI/MIT Press.

Jennings, N. R. (1999), Agent-based Computing: Promise and Perils. In: *Proceedings of 16th International Joint Conference on Artificial Intelligence (IJCAI-99*), Stockholm, Sweden, pp. 1429-1436.

Jennings, N.R. (2000), On Agent-Based Software Engineering, *Artificial Intelligence*, Vol. 117 (2), pp 277-296.

Lesser, V. (1999), Cooperative Multi-agent systems: A personal view of the state of the art, *In IEEE Transactions on Knowledge and Data Engineering*, Vol. 11, (1), pp 133-142.

Coutinho, L.R, Brandao, A.A.F, Sichman, J.S., Hübner, J.F. and Boissier, O. (2010), A model-based architecture for organizational interoperability in open multiagent systems. In Julian Padget et al., editor, *Coordination, Organizations, Institutions and Norms in Agent Systems V*, volume 6069 of *LNCS*, Spriner, pp 128-140.

Miles, R. E., C. C. Snow, J. A. Mathews, G. Miles and H. J. Coleman Jr. (1997), Organizing in the Knowledge Age: Anticipating the Cellular form, *Academy of Management Executive*, Vol. 11(4), pp. 7-20.

Nwana, H.S. (1996), Software Agents: An overview, *Knowledge Engineering Review*, Vol.11, (3), pp. 1-40.

Obonyo, E.A. Anumba C.J. and Thorpe, A. (2004), Specification and Procurement of Construction Products, A Case for an Agent-based System, *International Journal of IT in Architecture, Engineering and Construction*, Vol. 2 (3), , pp 204-215.

Obonyo, E.A. (2004), APRON: Agent-based specification and procurement of construction products, Doctoral Thesis, Department of Civil and Building Engineering, Loughborough University, UK.

Obonyo, E.A. Anumba C.J. and Thorpe, A. (2005a), APRON: An Agent-based Specification and Procurement System for Construction Products, *Engineering, Construction and Architectural Management*, Vol. 12 (4), pp 329-350.

Obonyo, E.A. Anumba C.J. and Thorpe, A. (2005b), Specification and Procurement of Construction Products Using Agents, In (Anumba C. J., Ugwu O. O & Ren Z (eds) *Agents and Multi-Agent Systems in Construction*, Taylor & Francis Group, London & New York, pp 186-210.

Omicini, A. (2001), SODA: Societies and Infrastructures in the Analysis and Design of Agent-Based Systems, In P. Ciancarini and M. Wooldridge (eds.), Agent-Oriented Software Engineering: First International Workshop, AOSE 2000, *Lecture Notes in Artificial Intelligence*, Vol. 1957. Springer-Verlag, Berlin Heidelberg, pp 185-194.

Pavón, J. and Gómez-Sanz, J. (2003), Agent Oriented Software Engineering with INGENIAS, *International Central and Eastern European Conference on Multi-Agent Systems*, Springer Verlag.

Padgham, L. and Winikoff, M. (2002), Prometheus: A Methodology for Developing Intelligent Agents, In F. Giunchiglia, J. Odell, and G. Weiss (eds), *Agent-Oriented Software Engineering III: Third International Workshop, AOSE 2002*, Revised Papers and Invited Contributions (Pt. 3), Springer Verlag, pp 37-38.

Parunak, H.V.D. and Odell, J. (2002), Representing Social Structure in UML, In *Agent-Oriented Software Engineering II*, Springer, pp 1-16.

Prietula, M., Carley, K., and Gasser, L. (ed) (1998), *Simulating Organizations: Computational Models of Institutions and Groups*, MIT Press Books, The MIT Press, edition 1, volume 1, number 026266108x, June.

Sierhuis, M., Jonker, C., van Riemsdijk, B., and Hindriks, K. (2009), Towards Organization Aware Agent-based Simulation, *International Journal of Intelligent Control and Systems*, Vol. 14 (1), pp 62-76.

Taveter, K. and Wagner, G. (2005), Towards Radical Agent-Oriented Software Engineering Processes Based on AOR Modelling, In Henderson-Sellers, B. and Giorgini, P. (eds.), *Agent-Oriented Methodologies*, Idea Publishing Group, UK, pp 277-316.

URL 1: http://jade.tilab.com/ (Accessed 15th September 2002)

URL2: http://www.philips.co.uk (Accessed 14th May 2001)

URL 3: http://agenttool.cis.ksu.edu/ (Accessed 10th December 2001)

URL 4: http://www.madkit.org (Accessed 9th June 2010)

Wooldridge, M., Jennings N. and Kinny, D. (2000), The Gaia Methodology for Agent-Oriented Analysis and Design, *Journal of Autonomous Agents and Multi-Agent Systems*, Vol. 3 (3), pp 285-312.

Yu, E. and Mylopoulos, J. (1995), From E-R to 'A-R' -- Modelling Strategic Actor Relationships for Business Process Reengineering, *International Journal of Intelligent and Cooperative Information Systems*, World Scientific Publishing, Vol. 4, (2/3), pp 125-144.

Zambonelli, F. Jennings, N. R. and Wooldridge, M. J. (2000), Organizational Abstractions for the Analysis and Design of Multi-Agent Systems, Workshop on Agent-oriented Software Engineering ICSE 2000.

Zambonelli, F., Jennings, N. R. and Wooldridge, M. J. (2001), Organizational Rules as an Abstraction for the Analysis and Design of Multi-Agent Systems, *International Journal of Software Engineering and Knowledge Engineering*, Vol.11 (3), pp. 303-328.

Zambonelli, F. and Parunak, H.V.D. (2002), From Design to Intentions: Signs of a Revolution, In *AAMAS 2002*, ACM Press, pp 455-456.

Zambonelli, F., Jennings, N. R. and Wooldridge, M. J. (2003) Developing Multi-agent Systems: The Gaia Methodology, *ACM Transactions on Software Engineering Methodology*, Vol. 12(3), pp 317 - 370.

BLANK PAGE