The Interoperability Problem: Bringing together Mobile Agents and Agent Communication Languages

Yannis Labrou, Tim Finin and Yun Peng
Computer Science and Electrical Engineering
University of Maryland Baltimore County
Baltimore MD 21250 USA
{jklabrou , finin , ypeng}@csee.umbc.edu

Abstract

Interoperability is a central issue for both the mobile agents community and the wider agents community. Unfortunately, the interoperability concerns are different between the two communities. As a result, inter-agent communication is an issue that has been addressed in a limited manner by the mobile agents community. Agent communication languages (ACLs) have been developed as tools with the capacity to integrate disparate sources of information and support interoperability but have achieved limited use by mobile agents. We investigate the origins of the differences of perspective on agent-to-agent communication, examine the reasons for the relative lack of interest in ACLs by mobile agents researchers and explore the integration of ACLs into mobile agents frameworks.

1. Introduction

There are two kinds of discussions that have plagued agent-related mailing lists and similar forums, over the past few years: (a) what is an agent, and (b) why do we need mobile agents. Enormous time and bandwidth have been spent only to leave the impartial observer with the conclusion that the answers to both questions mostly come down to matters of taste. The lurking provocateurs might suggest that it is the lack of convincing agent applications that makes these questions irrelevant to the practice of software development and prone to subjective philosophical musings. We will forego giving our perspective on these questions and will instead focus on another question, which much to our surprise, has not attracted much attention in such forums. Even though agent to agent communication is a central issue in the pursuit of interoperability in the agent community, why is it rarely the case that agent communication languages (ACLs) and mobile agents are mentioned in the same context?

We support the view of software agents [28] as an emerging software-building paradigm. Leaving the hype aside, the paradigm introduces a powerful and ubiquitous abstraction that exhibits the following key concepts: (a) a software agent is an autonomous goal-directed process capable of performing actions, (b) is situated in, is aware of and reacts to its environment, and (c) cooperates with other agents (software or human) to accomplish its tasks. In the context of the question we are asking, we adopt the viewpoint that suggest that mobile agents may be thought as programmed entities that can freely roam the network and act on behalf of their users [25]. A slightly more technical definition suggests that mobile agents are programs, typically written in a script language, which may be dispatched from a client computer and transported to a remote server computer for execution [19]. From a historical perspective, the work on distributed computing led to mobile agents following a route through the problem of process migration and the concept of mobile objects [25]. A large part of the work on agents, on the other hand, has its roots in various branches of Artificial Intelligence. We intend to further explore the different lineage between mobile agents and (just) agents.

If mobile agents are first and foremost agents and secondarily "mobile" artifacts, why have they remained untouched by the discourse on ACLs? We will start by examining, in Section 2, what interoperability has come to mean for the mobile agent community and the (largely non-mobile) agent community, respectively. We will then argue our case for ACL support for mobile agents in Sections 3 and 4; our argument is based on the power of an ACL as an interoperability mechanism and tool, and the observation that such interoperability is at least as important for mobile agents as it is for any other kind of agents. After a brief coverage of today's ACLs and their status in Section 5, we discuss reasons why the mobile agent community might be reluctant to fully engage ACLs and explore their integration in mobile agent frameworks and systems.

2. Agents and interoperability

Agents are meant to work with other agents. A central point of the agent paradigm of software development is that communities of agents are much more powerful than any individual agent, which immediately raises the necessity for interoperable agent systems. But the mobile agent community and the agent community at-large do not necessarily refer to the same things or are concerned about the same problems when talking about interoperability amongst agents. Before exploring the historical roots of these conceptual differences, we will try to identify the issues that each community associates with interoperability. Briefly, for the mobile agents community interoperability work focuses on the execution environment and the standardization of some of its aspects and features; in the case of (non-mobile) agents, there is no notion of an execution environment and the focus is on communication as the means for achieving interoperability. In the latter case, interoperability is akin to effectively exchanging the information and knowledge content of the agents.

2.1. Mobile agents and interoperability

Mobile agents reside in a highly heterogeneous environment; this heterogeneity presents itself in many dimensions. Mobile agents migrate to a host where an execution environment is set up for them; upon arriving there, they might execute code, make remote procedure calls (RPCs) in order to access the resources of the host, collect data and eventually might initiate another process of migration to another host. While residing on a particular host, mobile agents might have limited interaction (communication) with other agents on the host through an RPC-type mechanism. A potential problem arises from the fact that not all platforms for mobile agents are the same.

The Mobile Agent Facility (MAF) proposal [16], which was subsequently replaced by MASIF [23, 24], is an attempt to standardize some aspects of this execution environment. The proposal was still under investigation by the Object Management Group (OMG), as of the mid-1998. MASIF is a collection of definitions and interfaces that provides an interoperable interface for mobile agent systems.

MASIF's interoperability is not about language interoperability but instead it aims at interoperability between agent systems written in the same language although possibly by different vendors. MASIF focuses on standardizing three things: (1) agent management, *i.e.*, standard operations such as creating an agent, suspending it, resuming, and terminate it; (2) agent transfer, *i.e.*, a common infrastructure for agent applications to freely move among agent systems of different types; and finally (3) names for agents and agent systems. It is expected that the use of a standardized syntax and semantics of agent and agent system names will allow agent systems and agents to identify each other, as well as clients to identify agents and agent systems.

There are issues that are not addressed by MASIF, either because they are outside the scope or because the field is not mature enough for standardization over some issues. One of them is agent communication as it is extensively addressed by CORBA. As such a viewpoint suggests, agent communication for the mobile agents community means, at least most of the time, the exchange of objects or object references.

2.2. Agents and interoperability

The broader agent community is often plagued by a less focused view of agents. As a result its perspective on agent communication is marked by a desire for inclusiveness. Agent research ought to take into account the following realities: (a) various languages, representing different programming paradigms (procedural, object-oriented, logic, functional, etc.) will be used for implementing agents, (b) hardware platforms and operating systems are expected to be equally varied, and (c) agents are going to be written as autonomous applications and thus few assumptions might be made about their internal structure. The methodology used to address these problems emphasizes the identification of distinct layers that lead to an intuitive break-up of the larger problem:

One layer is that of translation between languages in the same family (or between families) of languages. This is a very formidable task. The Object Management Group (OMG) standardization effort is an example of work in this direction, within the family of object-oriented languages.

Another layer is concerned with guaranteeing that the semantic content of tokens is preserved among applications; in other words, the same concept, object, or entity has a uniform meaning across applications even if different names are used to refer to it. Every application incorporates some view of the domain (and the domain knowledge) it applies to. The technical term for this body of background knowledge is *ontology* [18].

More formally, an ontology is a particular conceptualization of a set of objects, concepts and other entities about which knowledge is expressed and of the relationships that hold among them. An ontology consists of terms, their definitions, and axioms relating them [17]; terms are normally organized in a taxonomy. For example, the fact that a book can be described with attributes such as its title, its ISBN and its authors names is a particular conceptualization of the concept book. This conceptualization exists and makes sense regardless of the presence of an application which employs it or of the particular strings one uses to encode the `title`, `ISBN#` and `author` attributes. A slightly different conceptualization that adds a selling price for each book might extend the previous conceptualization but this extended version should still make sense to an application that is not concerned with price.

A third layer relates to the communication between agents. This is not about transporting bits and bytes between agents; agents should be able to communicate complex attitudes about their information and knowledge content. Agents need to ask other agents, to inform them, to request their services for a task, to find other agents who can assist them, to monitor values and objects, and so on. Such functionality, in an open environment, can not be provided by a simple remote procedure call mechanism.

3. A wider interoperability perspective

So, the first question to ask is what happens when a mobile agent needs to interact with mobile agents that are currently hosted in a platform of a different design; in such a case a mobile agent can not even visit such a platform. Hopefully, a wide acceptance of the MAF/MASIF proposal will help with this problem. But, let us consider the case of a mobile agent that needs to interact with some agent that is not endowed with mobility. In this case, there is no agent platform to visit. Should the mobile agent be prevented from interacting with the static agent? Yet another awkward case might arise when a mobile agent needs to interact with an information source that is not a full fledged agent but might offer some way of interacting with its information content. Nevertheless, regardless of these discrepancies, it will be necessary for all these disparate sources of information content to seamlessly interact with one another. Interaction means more than simply exchanging messages; it also involves facilities for finding these information sources. Finding, means not only physically locating them on the network, but also been able to identify them or judge them relevant based on their apparent content or capabilities. And even in a world where everything is an *object*, finding an object does not tell you what the object is about.

A whole different (qualitatively) range of problems appears at a different level. These potential conversants for the mobile agents, *i.e.*, mobile agents implemented for different platforms, static agents and non-agent information sources, may be implemented or make use of languages different from the mobile agent's one. Furthermore, the tokens they use to refer to the same concepts, entities and objects might differ; one agent might refer to the title of a book as `book_title` and another might use the token `title`, but we would like to think of both as referring to the same concept.

There is, finally, another issue of a different nature, which is particular to mobile agents. Mobile agents carry their own code that prescribes the *"how to"* of the tasks they can tackle. This procedural approach towards problem-solving can be very limiting. In the current mobile agents paradigm there is not much room for a declarative approach, *i.e.*, one in which agents simply specify the task they want performed, leaving the details of carrying out the task to the recipient of the request. The procedural vs. declarative debate can be a particularly lengthy one; regardless of where one stands on this issue, it is certain that the declarative approach requires an amendment in the current conceptualization of mobile agents. In this revised framework, mobile agents ought to be able to communicate tasks in a common language. The mobile agent paradigm hints to an expectation of all information sources been accessible through procedural means, which we find unrealistic. To address all of these issues the community has suggested a layered abstraction that treats agents at a higher level than the details of their internal structure.

4. ACLs as an interoperability mechanism

One common abstraction of an agent is as a *Virtual Knowledge Base* [13, 15] – a collection of (mostly) declarative information and knowledge and an associated inference mechanism that allows it to proactively make inferences, answer queries, and (perhaps) take action. In this context interoperability is often interpreted as a problem of enabling agents with sharing their information and knowledge content.

Three basic problems need to be addressed for agents to effectively share knowledge. First, how we can translate from one knowledge representation language to another; second, how we can guarantee that the meaning of concepts, objects, relationships is the same across different agents; and third, how this potentially sharable knowledge is going to actually be shared, communicated between agents.

An Agent Communication Language (ACL) is a tool that follows the path of this layered abstraction of the interoperability question. An ACL can be thought of as a collection of message types each with a reserved meaning. A communication language is not concerned with the physical exchange, over the network, of an expression in some language, but rather with stating an attitude about the content of this exchange.

Therefore, for example, a communication language can distinguish between a particular message exchange that suggests a query, an interest for a particular content, a promise or commitment to perform a future action, or an assertion.

From a software engineering point of view, an agent communication language can be viewed as another messaging protocol, but with two major differences: (a) it describes the application and the actions it can perform, or it can be requested to perform, at a higher level of abstraction, and (b) offers a larger variety of message types. An ACL thus introduces a powerful abstraction because it separates (1) the expressions that are the content of the exchange and (2) their meaning, from the attitude that is expressed about them.

ACLs offer a conceptual framework that can assist us in addressing the difficult problem of achieving interoperability between applications. Although interoperability is not a problem unique to agents, the very fact that software agents are meant to be, almost by definition, autonomous applications designed with minimal a priori expectations for the state of the rest of the universe brings interoperability to the forefront. Mobile agents are even more susceptible to the perils of heterogeneity.

An agent communication language offers three kinds of advantages.

First, an ACL supports interoperatiblity between static and mobile agents, between mobile agents designed for different agents platforms, and also between mobile agents and static agentified information sources. Second, the declarative nature of most ACLs provides many features that make interoperability easier, such as abstracting away some of the lower-level, more procedural aspects of the systems involved. Finally, the higher level of abstraction at which ACLs operate can accommodate multiple paradigms.

Thus far, most of the work on mobile agents has focused on the problem of designing the agent platform and addressing major issues surrounding its design, such as security and authentication. Agent communication has been put in the back-burner, partly because of the concentration on fundamental issues and partly because the ACL alludes to a declarative approach that contradicts the procedural approach implicit in the mobile agent paradigm. There are some exceptions to this situation, of course, such as some experimental work in integrating Agent Tcl and KQML [32].

5. Current ACLs

The Knowledge Sharing Effort [26, 29] (KSE) was initiated circa 1990 by DARPA and it enjoyed the participation of dozens of researchers from both academia and industry. Its goal was to develop techniques, methodologies and software tools for *knowledge sharing and knowledge reuse*, at *design*, *implementation*, or *execution* time. The central concept of the KSE was that knowledge sharing requires communication, which in turn, requires a common language; the KSE focused on defining that common language. In the KSE model, software systems are viewed as (virtual) knowledge bases that exchange propositions using a language that expresses various complex attitudes [1] about these propositions.

Although originally agents were not part of the KSE vocabulary the conceptual break-down of the "common language problem" is applicable to what we currently refer to as agents. Expressions in a given agent's native language should be understood by some other agent that uses a different implementation language and domain assumptions. So, the first layer is that of (syntactic) translation between languages in the same family (or between families) of languages [2]. An other layer is concerned with guaranteeing that the semantic content of tokens is preserved among applications; in other words, the same concept, object, or entity has a uniform meaning across applications even if different "names" are used to refer to it. Every agent incorporates some view of the domain (and the domain knowledge) it applies to. The technical term for this body of "background" knowledge is *ontology*. More formally, an ontology is a particular conceptualization of a set of objects, concepts and other entities about which knowledge is expressed and of the relationships that hold among them. An ontology consists of terms, their definitions, and axioms relating them [17]; terms are normally organized in a taxonomy.

A final layer addresses the communication between agents. This is not about transporting bits and bytes between agents; agents should be able to communicate complex "attitudes" about their information and knowledge content. Agents need to ask other agents, to inform them, to request their services for a task, to find other agents who can assist them, to monitor values and objects, and so on. Such functionality, in an open environment, can not be provided by a simple RPC mechanism. Agents issue requests by specifying not a procedure but a desired state in a declarative language, *i.e.*, in some appropriate agent communication language.

Within the KSE, these layers were viewed as independent of another. The ACL is only concerned with capturing propositional attitudes, regardless of how propositions are expressed. But still, propositions are what agents will be "talking" about. KIF [14], a particular logic language, was proposed within the KSE as a standard to use to describe things within computer systems, *e.g.*, expert systems, databases, intelligent agents, etc. Moreover, it was specifically designed, within the context of the KSE, to make it

[1] The proper term is *propositional attitudes*. Propositional attitudes are three-part relationships between: (1) an agent, (2) a content-bearing proposition (*e.g.*, "it is raining"), and (3) a finite set of propositional attitudes an agent might have with respect to the proposition (*e.g.*, believing, asserting, fearing, wondering, hoping, being interested in, *etc.*). For example, $< a, fear, raining(t_{now}) >$.

[2] The Object Management Group (OMG) standardization effort is an example of work in this direction, within the family of object-oriented languages.

useful as an "interlingua". By this we mean a language which is useful as a mediator in the translation of other languages. KIF is a prefix version of first order predicate calculus with extensions to support non-monotonic reasoning and definitions. The language description includes both a specification for its syntax and one for its semantics. Ontolingua [11] and a variety of supporting tools, was the KSE "solution" to the problem of developing and maintaining ontologies. Researchers at Stanford's Knowledge Systems Laboratory have developed a set of tools and services to support the process of achieving consensus on common shared ontologies by geographically distributed groups. These tools are built around Ontolingua, a language designed for describing ontologies with it, and make use of the world-wide web to enable wide access and provide users with the ability to publish, browse, create, and edit ontologies stored on an ontology server. Users can quickly assemble a new ontology from a library of existing modules, extend the result with new definitions and constraints, check for logical consistency, and publish the result back to the library.

KQML [1, 20] illustrates the basic concepts of existing ACLs. All KQML dialects and FIPA ACL follow the same basic concepts of KQML that we discuss here. KQML is a high-level, message-oriented communication language and protocol for information exchange independent of content syntax and applicable ontology. So, KQML is independent of the transport mechanism (TCP/IP, SMTP, IIOP, *etc.*), independent of the content language (KIF, SQL, STEP, Prolog, *etc.*) and independent of the ontology assumed by the content.

5.1. KQML and ACL concepts

The KQML language is divided into three layers: the content layer, the message layer, and the communication layer. The content layer bears the actual content of the message, in the programs own representation language. KQML can carry any representation language, including languages expressed as ASCII strings and those expressed using a binary notation. Every KQML implementation ignores the content portion of the message, except to determine where it ends. The communication level encodes a set of features to the message which describe the lower level communication parameters, such as the identity of the sender and recipient, and a unique identifier associated with the communication. It is the message layer that is used to encode a message that one application would like to transmit to another. The message layer forms the core of the KQML language, and determines the kinds of interactions one can have with a KQML-speaking agent. The primary function of the message layer is to identify the network protocol to be used to deliver the message and to supply a speech act or performative which the sender attaches to the content (such as that it is an assertion, a query, a command, or any of a set of known *performatives*). In addition, since the content is opaque to KQML, this layer also includes optional features which describe the content language, the ontology it assumes, and some type of description of the content, such as a descriptor naming a topic within the ontology. These features make it possible for KQML implementations to analyze, route and properly deliver messages even though their content is inaccessible.

The syntax of KQML is based on the familiar *s-expression* used in Lisp, *i.e.*, a balanced parenthesis list. The initial element of the list is the performative; the remaining elements are the performative's arguments as keyword/value pairs. Because the language is relatively simple, the actual syntax is not significant and can be changed if necessary in the future. The syntax reveals the roots of the initial implementations, which were done in Common Lisp; it has turned out to be quite flexible. A KQML message from agent *joe* representing a query about the price of a share of IBM stock might be encoded as:

```
(ask-one
  :sender joe
  :content (PRICE IBM ?price)
  :receiver stock-server
  :reply-with ibm-stock
  :language LPROLOG
  :ontology NYSE-TICKS)
```

In this message, the KQML performative is *ask-one*, the content is *(price ibm ?price)*, the ontology assumed by the query is identified by the token *nyse-ticks*, the receiver of the message is to be a server identified as *stock-server* and the query is written in a language called *LPROLOG*. The value of the `:content` keyword is the content level, the values of the `:reply-with`, `:sender`, `:receiver` keywords form the communication layer and the performative name, with the `:language` and `:ontology` form the message layer. In due time, *stock-server* might send to *joe* the following KQML message:

```
(tell
  :sender stock-server
  :content (PRICE IBM 14)
  :receiver joe
  :in-reply-to ibm-stock
  :language LPROLOG
  :ontology NYSE-TICKS)
```

Though there is a predefined set of reserved performatives, it is neither a minimal required set nor a closed one. A KQML agent may choose to handle only a few (perhaps one or two) performatives. The set is extensible; a community of agents may choose to use additional performatives if they agree on their interpretation and the protocol associated with each. However, an implementation that chooses to implement one of the reserved performatives must implement it in the standard way.

One of the design criteria for KQML was to produce a language that could support a wide variety of interesting agent architectures. Thus, KQML introduces a small number of KQML performatives which are used by agents to describe the meta-data specifying the information requirements and capabilities; KQML also introduces a special class of agents called *communication facilitators* [15, 10, 22]. A facilitator is an agent that performs various useful communication services, *e.g.* maintaining a registry of service names, forwarding messages to named services, routing messages based on content, providing "matchmaking" between information providers and clients, and providing mediation and translation services.

5.2. FIPA and its ACL

Although the Knowledge Sharing Effort (KSE) introduced the major research issues and approaches to the interoperability problems of the agents community it did not offer a disciplined way for developing standards. This void was filled in 1996 by the Foundation for Intelligent Physical Agents (FIPA). FIPA is a non-profit association whose purpose is to "promote the success of emerging agent-based applications, services and equipment." FIPA's goal is to make available specifications that maximize interoperability across agent-based systems. FIPA operates through the open international collaboration of member organizations, which are companies and universities that are active in the field. The participating companies cover an impressive range of telco's and telecommunications equipment companies from Europe, the Far East, and to a lesser degree, from North America.

FIPA's operations are built around annual rounds of specification deliverables. Current specification is FIPA97 and can be found at the FIPA home-page [3] The *modus operandi* of FIPA is to assign tasks to Technical Committees (TCs) that have the primary responsibility for producing, maintaining and updating the specification(s) that are applicable to their task(s). A TC was assigned with the task of producing a specification for an Agent Communication Language. Along with the TC in charge of Agent Management (agent services, such as facilitation, registration and agent platforms) and Agent/Software Interaction (integration of agents with legacy software applications) they form the backbone of the FIPA specifications.

The FIPA Agent Communication Language (FIPA ACL), like KQML, is based on speech act theory: messages are actions, or communicative acts, as they are intended to perform some action by virtue of being sent. The specification consists of a set of message types and the description of their pragmatics, that is the effects on the mental attitudes of the sender and receiver agents. Every communicative act is described with both a narrative form and a formal semantics based on modal logic. The specification also provides the normative description of a set of high-level interaction protocols, including requesting an action, contract net and several kinds of auctions. The FIPA ACL is superficially similar to KQML. Its syntax is identical to that of KQML's except for the different names for some reserved primitives. Thus, it maintains the KQML approach of separating the outer "language" that defines the intended meaning of the message and the inner language, or "content language" that denotes the expression towards which the beliefs, desires and intentions of the interlocutors, as described by the meaning of the communication primitive, apply. The FIPA ACL specification document claims that FIPA ACL (like KQML) does not make any commitment to a particular content language. Although such a claim holds true for most primitives, there are FIPA ACL primitives for which some understanding of the language SL (Semantic Language) is necessary for the receiving agent to understand and process the primitive.

6. Bringing the two worlds together

We want to investigate the possibilities of supplementing mobile agents with the ability to handle an ACL. As we previously argued, doing so will make a new range of interoperability options available to mobile agents. The deep division between stationary and mobile agents is unnecessary and integrating ACLs into mobile agents frameworks might not be as hard of a problem as it often seems. Moreover, the MASIF proposal and the enduring presence of FIPA, along with the continuous interaction between the communities involved in these two endeavors, make the time right for the relatively minor adjustments needed to bring the two worlds together.

We would like to first trace the origins of the different perspectives between the agents and the mobile agents communities; we suggest that a convergence might be subtly taking place, at least at the conceptual level (Section 6.1). We then explore the particular technical areas that need to be addressed in order to achieve the integration of the two universes (Section 6.2.)

6.1. Origins of the different perspectives

The semantic approaches of current ACLs [21, 33, 7, 34] suggest the origins of much of the past and ongoing re-

[3] which was `http://drogo.cselt.stet.it/fipa` at the writing of this paper; a new official FIPA page is under construction, at `http://www.fipa.org`

search on ACLs. Rooted in various branches of Artificial Intelligence, these semantic approaches rely on multimodal logics that are often non-computable and/or have no efficient implementation. Despite these problems, BDI agents[4] [31, 12, 30] can be implemented with traditional AI languages, but we have relatively little experience in doing so with object-oriented languages like Java[5]. Mobile agents, however, are usually programmed in object-oriented or scripting languages. Traditionally, this is not the domain of AI programming languages and techniques nor do such languages usually contain libraries that offer the components typical of BDI agents, such as inference engines and knowledge representation libraries.

But things seem to be changing. An informal survey of the various multi-agent systems that use an ACL for interagent communication would reveal two interesting trends.

- Java is rapidly becoming the language of choice for building agents and knowledge based systems in general [6].

- Many of the new APIs for agent communication languages [3, 27, 9, 8] offer support for modeling, manipulating and reasoning about conversations among agents

Conversations offer an intuitive way to structure an agent's activities. Also, given the problematic nature of compliance with the ACL's semantic account, conversations shift the focus from the internals of the agent to its observable behavior expressed as sequences of messages sent to other agents. Agents can agree on a conversation protocol for a particular task (e.g., negotiation or auction) and then engage in a scripted interaction. We do not suggest that this is a conformance test, but it might be useful for an agent designer to know that its interlocutors engage in a scripted, pre-specified communicative

The reluctance of the mobile agents community to engage into the world of ACLs has to do, in some degree, with the programming languages that dominate each branch of the agents paradigm. The fact that Java is becoming the favorite language for programming agents of all types, however, might be changing that. The reality of a language like Java can be seen at the realm of the semantic definition of ACLs.

This last problem though, applies to any agent mobile or not. The realization of the limits of these semantic accounts in terms of software development can explain, at least in part, the interest in conversations.

[4] The aforementioned semantic approaches rely heavily on modalities such belief, desire and intention. BDI agents, *i.e.*, Belief, Desire, Intention Agents have concrete, computational implementations of these modalities.

[5] For some very current work along these lines, see [4, 5] for a description a Java-based framework for BDI agents.

6.2. Integrating ACL's and mobile agents

From a software design point of view, for an agent system to "speak" KQML (or FIPA ACL for that matter) the following things have to be provided:

1. a suite of APIs that facilitate the composition, sending and receiving of ACL messages,

2. an infrastructure of services that assist agents with naming, registration and basic facilitation services (finding other agents that can do things for your agent), and

3. the code that for every reserved message type (performative or communicative act) takes the actions prescribed by the semantics, for the particular application; this code depends on the application language, the domain and the details of the agent system that uses the ACL.

Normally as a programmer you would only have to provide item (3). Items (1) and (2) should be re-usable components that one can just integrate into the application code; actually item (2) does not even need to be integrated because it ought to be provided as continuous running services that a new agent can just use. Unfortunately, such services were not the focus of the standardization efforts until very recently. There is no service where one can register an agent by just sending a registration message. This problem, which is being addressed by the agents community, has to do primarily with the lack of agreement on the naming scheme for agents. Providing the code (item (3)) that processes the primitives is more of an art than a science.

Because mobile agents are typically programmed in object-oriented or scripting language, it might seem that the major obstacle for adopting ACLs in the mobile agents community is the difficulty of translating the semantics account into code in such languages that complies with the ACL's specification. As Java rapidly becomes the language of choice in the agents world, we feel that this is not the essential problem, or at least, it is not more of a problem for mobile agents than it is for agents in general. This view seems to be supported by the emphasis in conversations and the specification of conversation protocols; this shift of focus suggests a programming-friendly way to account for an agent's communicative behavior.

Instead, we identify two major technical obstacles that ought to be addressed for the integration of mobile agents and the ACL infrastructure.

Th first problem is that mobile agents ought to be able to compose, send and receive ACL messages. At this point, we are only concerned with the code necessary for these tasks. Our answer to this issue is that mobile agents need not actually have such code; the *agent place* [6] can provide this functionality. We do not suggest that this kind of interaction has to be specified, in the MASIF proposal. Implementation of places for agent systems can provide such functionality. As we mentioned, there is a variety of Java APIs that support that kind of functionality; the only issue is integrating them with places' implementations. The mobile agent will use its usual mechanisms of interacting with the place in order to retrieve its messages or to submit messages for sending. It is likely that a mobile agent will only retrieve or submit the content of the ACL message, *i.e.*, the value of the `:content` parameter, leaving the place's services with the task of composing a complete ACL message and parsing and extracting its `:content`.

A place can interact with other places, facilitators and ontology services in order to manage delivery to the appropriate recipient, or language and ontology translation issues. But these aspects of interoperability and the necessary infrastructure to achieve it, do not differ from the requirements for non-mobile agents. Of course mobile agents are still left with the task of processing the content of an ACL message, as they see fit. This task, as is the case for non-mobile agents, will depend on the particulars of the agent and its domain. The only real difference is that the compositions, sending and receiving of messages is delegated to the agent place.

A second obstacle is that mobile agents do not have a permanent location on the network. This has repercussions for the naming scheme needed to identify and refer to them in the network. Technically speaking, the naming scheme and its implicit ontology is not part of an ACL specification. Agent naming though is viewed as one of the most essential agent management issues that the ACL community is concerned with. Over time, a variety of suggestions have been made, and FIPA is exploring it in the context of its *Agent Management* Technical Committee. The views of the agents community have been gradually shifting from the earlier viewpoint of a permanent fixed agent location and a symbolic name associated with it, to a more flexible view, where agents may or may not be connected to the network all the time, or may suspend their operation and resume it in some new location or even might have multiple names associated with a unique identity (in such a case these name are related to each other and to the common identity). Such a view seems to include, at least conceptually, the concept of a mobile agent that changes locations carrying (all or parts of) state and data. To the extend that such concepts are reflected in current naming mechanisms and conventions its should be possible to find a common ground between the two communities. It would be fruitless to get into more detail about naming matters since existing proposals are just proposal and bound to change, but this is exactly the opportunity we ought to take advantage of.

If we were to assign priority to the technical issues mentioned here, the most important concern would be to ensure that naming conventions are consistent between the MASIF proposal and whatever gets adopted in the wider agents community (for example FIPA's choices).

7. Conclusions

Software agents offer a new paradigm for building very large scale distributed heterogeneous applications that focus on the interactions of autonomous, cooperating processes that can adapt to humans and other agents. As such, communication is key to realizing the potential of this new paradigm, just as the development of human language was critical to the development human society and culture. This holds for mobile agents as well as for stationary ones. Agents use an Agent Communication Language to communicate information and knowledge. Some[15] even go so far as to suggest that the use of a rich ACL is a defining characteristic of a software agent, *i.e.*, a software agent is any process that makes appropriate use an ACL to exchange information. Although this sounds circular, it is not if we stipulate that an ACL should be sufficiently rich to encode a wide range of knowledge and a reasonable set of propositional attitudes toward sentences in the ACL.

The distinguishing characteristic of mobile agents, in terms of being agents, should not be their ability to move around the network but their ability to act (autonomously) on behalf of their users and eventually perform tasks for them. Mobility is orthogonal to communication; what is important is that both functionalities might be ways to perform tasks and to interoperate and collaborate with other agent systems. Agents can achieve high-level interoperability by communication at a higher-level of abstraction involving such concepts as beliefs, goals, expectations and intentions. Currently, mobile agent present a very narrow view of agent communication that does not take full advantage of communication as an interoperability mechanism. We argue that at this critical period of standardization efforts in the broader area of agents, bringing mobile agents and agent communication together is an opportunity not to be missed. Doing so, entails relatively small adjustments primarily in agent naming schemes. Having achieved that, a mobile agent can "speak" an agent communication language, having an agent platform manage the details of handling ACL messages for it.

References

[1] ARPA Knowledge Sharing Initiative. Specification of the

[6] A place is a context in which an agent executes. It is associated with a location, which consists of the place name and the address of the agent system where the place resides [35]

KQML agent-communication language. ARPA Knowledge Sharing Initiative, External Interfaces Working Group, July 1993.

[2] J. M. Bradshaw, editor. *Software Agents*. AAAI/MIT Press, 1995.

[3] J. M. Bradshaw, S. Dutfield, P. Benoit, and J. D. Woolley. Kaos: Toward an industrial-strength open agent architecture. In J. M. Bradshaw, editor, *Software Agents*. AAAI/MIT Press, 1997.

[4] P. Busetta and R. Kotagiri. The bdim agent toolkit design. Technical Report 97/15, Department of Computer Science, The University of Melbourne, Australia, 1997.

[5] P. Busetta and R. Kotagiri. An architecture for mobile bdi agents. In J. Carroll, G. B. Lamont, D. Oppenheim, K. M. George, and B. Bryant, editors, *Proceeding of the 1998 ACM Symposium on Applied Computing (SAC'98)*, pages 445–452, February 1998.

[6] V. K. Chaudhri, A. Farquhar, R. Fikes, P. D. Karp, and J. P. Rice. Okbc: A programmatic foundation for knowledge base interoperability. In *Proceedings of the National Conference on Artificial Intelligence (AAAI98)*. AAAI Press, 1998.

[7] P. R. Cohen and H. Levesque. Communicative actions for artificial agents. In *Proceedings of the 1st International Conference on Multi-Agent Systems (ICMAS'95)*. AAAI Press, June 1995.

[8] R. S. Cost, T. Finin, Y. Labrou, X. Luan, Y. Peng, I. Soboroff, J. Mayfield, and A. Boughannam. Jackal: a java-based tool for agent development. In *Working Papers or the AAAI-98 Workshop on Software Tools for Developing Agents*, august 1998.

[9] R. S. Cost, I. Soboroff, J. Lakhani, T. Finin, and E. Miller. TKQML: A scripting tool for building agents. In M. Wooldridge, M. Singh, and A. Rao, editors, *Intelligent Agents Volume IV – Proceedings of the 1997 Workshop on Agent Theories, Architectures and Languages*, volume 1365 of *lnai*, pages 336–340. sv, Berlin, 1997.

[10] K. Decker, K. Sycara, and M. Williamson. Middle-agents for the internet. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, Nagoya, Japan, August 1997.

[11] A. Farquhar, R. Fikes, and J. Rice. The ontolingua server: A tool for collaborative ontology construction. In *KAW96*, November 1996.

[12] K. Fischer, J. P. Müller, and M. Pischel. A pragmatic BDI architecture. In M. Wooldridge, J. P. Müller, and M. Tambe, editors, *Intelligent Agents Volume II—Proceedings of the 1995 Workshop on Agent Theories, Architectures, and Languages (ATAL–95)*, Lecture Notes in Artificial Intelligence. Springer–Verlag, 1996. (In this volume).

[13] M. Genesereth. Designworld. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 2,785–2,788. IEEE CS Press.

[14] M. R. Genesereth. Knowledge interchange format version 3.0 reference manual. Technical Report Logic Group Report Logic-92-1, Stanford University, June 1992.

[15] M. R. Genesereth and S. P. Katchpel. Software agents. *CACM*, 37(7):48–53, 147, 1994.

[16] O. M. Group. Mobile agent facility (joint submission). Technical report, Object Management Group, 1997.

[17] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 2:199–220, 1993.

[18] N. Guarino. The ontological level. In R. Casati, B. Smith, and G. White, editors, *Philosophy and the Cognitive Sciences*. Hölder-Pichler-Tempsky, Vienna, 1994.

[19] C. G. Harrison, D. Chess, and A. Kershenbaum. Mobile agents: Are they a good idea? Research report, T.J. Watson Research Center, 1994.

[20] Y. Labrou and T. Finin. A proposal for a new kqml specification. Technical Report Technical Report TR-CS-97-03, University of Maryland Baltimore County, 1997.

[21] Y. Labrou and T. Finin. Semantics and conversations for an agent communication language. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*, Nagoya, Japan, August 1997.

[22] D. L. Martin, H. Oohama, D. Moran, and A. Cheyer. Information brokering in an agent architecture. In *Proceedings of the Second International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, London, Apr. 1997. The Practical Application Company Ltd.

[23] D. Milojicic, M. Breugst, I. Busse, J. Campbell, S. Covaci, B. Friedman, K. Kosaka, D. Lange, K. Ono, M. Oshima, C. Tham, S. Virdhagriswaran, and J. White. Masif: The omg mobile agent system interoperability facility. *Personal Technologies*, 2(3), December 1999.

[24] D. Milojicic, M. Breugst, I. Busse, J. Campbell, S. Covaci, B. Friedman, K. Kosaka, D. Lange, K. Ono, M. Oshima, C. Tham, S. Virdhagriswaran, and J. White. Masif: The omg mobile agent system interoperability facility. In K. Rothermel and F. Hohl, editors, *Mobile Agents, Proceedings of the Second International Workshop, MA'98*, LNCS 1477. Springer-Verlag, 1998.

[25] D. S. Milojicic, M. Condict, F. Reynolds, D. Bolinger, and P. Dale. Mobile objects and agents. In *"Distributed Object Computing on the Internet" AdvancedTopics Workshop, Second USENIX Conference on Object Oriented Technologies and Systems (COOTS)*, Toronto, Canada, 1996.

[26] R. Neches, R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator, and W. Swartout. Enabling technology for knowledge sharing. *AI Magazine*, 12(3):36–56, Fall, 1991.

[27] M. H. Nodine and A. Unruh. Facilitating open communication in agent systems: the InfoSleuth infrastructure. In M. Singh, A. Rao, and M. Woolridge, editors, *Proceedings of the 14th Annual Workshop on Agent Theories, Architectures and Languages (ATAL '97)*, Providence, RI, 1997.

[28] H. S. Nwana. Software agents: an overview. *Knowledge Engineering Review*, 11(3):1–40, September 1996.

[29] R. S. Patil, R. E. Fikes, P. F. Patel-Schneider, D. McKay, T. Finin, T. Gruber, and R. Neches. The darpa knowledge sharing effort: Progress report. In M. Huhns and M. Singh, editors, *Readings in Agents*. Morgan Kaufmann Publishers, 1997. (reprint of KR-92 paper).

[30] A. S. Rao. Decision procedures for propositional linear–time belief–desire–intention logics. In M. Wooldridge, J. P. Müller, and M. Tambe, editors, *Intelligent Agents Volume II—Proceedings of the 1995 Workshop on Agent Theories, Architectures, and Languages (ATAL–95)*, Lecture Notes in

Artificial Intelligence. Springer–Verlag, 1996. (In this volume).

[31] A. S. Rao and M. P. Georgeff. Modeling rational agents within a bdi-architecture. In J. Allen, R. Fikes, and E. Sandewall, editors, *Principles of Knowledge Representation and Reasoning: Proc. of the Second International Conference (KR'91)*, pages 473–484. Kaufmann, San Mateo, CA, 1991.

[32] D. Rus, R. Gray, and D. Kotz. Transportable information agents. In *1997 International Conference on Autonomous Agents*, Marina del Ray, CA, 1997.

[33] M. Sadek. A study in the logic of intention. In *Proceedings of the 3rd Conference on Principles of Knowledge Representation and Reasoning (KR'92)*, pages 462–473, Cambridge, MA, 1992.

[34] I. A. Smith and P. R. Cohen. Toward a semantics for an agent communications language based on speech-acts. In *Proceedings of the 13th National Conference on Artificial Intelligence*. AAAI/MIT Press, August, 1996.

[35] J. White. Mobile agents. In J. M. Bradshaw, editor, *Software Agents*. MIT Press, 1995.