BESA: Behavior-oriented, Event-driven and Social-based Agent Framework

Enrique González, Jamir Avila, César Bustacara

Pontificia Universidad Javeriana
Facultad de Ingeniería
Departamento de Ingeniería de Sistemas

Postal address: Carrera 7 40-62, Bogotá, Colombia
Telephone: (571)3208320 ext. 5308
Fax: (571)3208320 ext. 5338
Email: egonzal@javeriana.edu.co
BESA: Behavior-oriented, Event-driven and Social-based Agent Framework

Enrique González, Jamir Avila, César Bustacara
Pontificia Universidad Javeriana - Facultad de Ingeniería
Grupos de Investigación SIDRE-SIRP
Bogotá, Colombia

Abstract

This paper presents the BESA framework, which is intended to provide a flexible model that supports the design and implementation of MultiAgent Systems (MAS). The abstract model of BESA is based in three fundamental concepts: an event-driven control approach implementing a select like mechanism, a modular behavior-oriented agent architecture, and a social-based support for cooperation between agents. The fusion of these concepts into a parallel architecture is the main innovation of BESA.

The BESA architecture is composed of three levels: agent level, social level and system level. The internal architecture of an agent integrates two important features: a modular composition of behaviors and an event selector mechanism. In the BESA abstract model, intermediate levels of abstraction are incorporated allowing to define and handle hierarchical organizations. An agent system implemented using the BESA architecture is seen as a distributed system composed of one or several BESA-containers. The main components of the BESA framework have been implemented using the Java language; its correct operation has been proven by building a multi-robot simulator.

Keywords: MultiAgent Systems, Parallel Architecture, Intelligent Agents, Agent Framework

1 Introduction

Actually, Agents and MultiAgent Systems (MAS) are one of the most prominent and attractive technologies in Engineering and Computer Science. Agent and MAS technologies, methods, and theories are currently contributing to many diverse domains such as information retrieval, user interface design, robotics, computer games, education and training, smart environments, social simulation, management projects, e-business, knowledge management, virtual reality, etc [1][2][3]. They are not only a very promising technology, but also they are emerging as a new way of thinking: a conceptual paradigm for analyzing problems and for designing systems, a mean to deal with complexity, distribution and interactivity, and perhaps a new perspective on computing and intelligence.

An Agent is an entity that includes mechanisms to receive perceptions from its environment and modify it. The work of an agent is to decide or to infer which is the most adequate action to achieve a specific goal. An agent has several resources and skills, and frequently it can communicate with other agents [4]. The correct action is selected using a function mapping that can be expressed in different ways, ranging from simple condition-action rules to complex inference mechanisms. In some cases the mapping function can be given, in agents with mayor autonomy this function can be directly learned by the agent.
The capabilities of an isolated agent are limited to its resources and abilities. When objectives get more complex, the mapping function to select the best action is less efficient, because the complexity of this function is increased. Thus, it is more efficient to build several agents, where each agent contributes to achieve the general goal. A MAS can be defined as a collection of agents that cooperate to achieve a goal [5].

A MAS contains a set of heterogeneous agents running on a platform that support service discovery and agent interaction. In a most extended context, a MAS Framework must consider the following topics: communication between agents to exchange information or knowledge, resource allocation [6][7]. In a MAS, the agents must be autonomous both at design time and at runtime, adaptive, and cooperative. This approach allows building complex systems in a distributed and heterogeneous world.

Agent communication requires an standard language that permits them to interact while hiding the details of their internal operation, generating agent communities that tackle problems that no individual agent could solve. An agent communication language (ACL) is only concerned with capturing propositional attitudes regardless of how propositions are expressed [8]. KQML is a high-level, message-oriented communication language and protocol independent of content syntax and applicable ontology. FIPA ACL is an agent communication language, 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 FIPA ACL specification consists of a set of performatives and the description of their intentions, that is, the effects of the mental attitudes of the sender and receiver agents. The specification describes every communicative act with both a narrative form and a formal semantics based on modal logic [8][9].

There are several platforms and architectures to develop agent applications. The Genesereth architecture with emphasis on interoperability, Rosenschein architecture facilitates negotiation between agents, Wiederhold architecture developed the mediator concept, the Cohen architecture emphasizes that a system must be open and extensible [2][10]. More recently, the FIPA standard has evolved, and in consequence several platforms have emerged that comply with it [9]. In particular, several platforms have been developed using the Java language programming, because its portability to different operating systems and hardware. Some of the most significant platforms are JADE, ZEUS, FIPA-OS and April [11][12].

In this paper the BESA model is introduced. The main innovation of this architecture is the fusion into a parallel architecture of the behavior-oriented construction of agents, the select mechanism of concurrent programming and the organization approach to design MAS. The framework aims to fulfill two main requirements. First, it is FIPA compliant, which means that it includes the elements that allow a SMA to inter-operate with other systems. Second, the design of a MAS is based in a distributed and concurrent approach.

In the next section, the fundamental underlying concepts of the BESA abstract model are introduced. Then, the three layers of the architecture are presented. Finally, the main implementation issues are outlined.

2 BESA Fundamental Concepts

The abstract model of BESA is based in three fundamental concepts: a modular behavior-oriented agent architecture, an event-driven control approach implementing a select like mechanism, and a social-based support for cooperation between agents.

2.1 Behavior-Oriented

When building agents, one of the critical problems to solve is the complexity; as the agent is intended to be more rational and autonomous, the elements involved became more complex. In order to deal with this growing problem, different modular architectures have been proposed. The fundamental idea is to break down a complex entity into a set of small simpler ones. This process can even done in a systematic way, for instance the methodology
presented by Foster to design concurrent systems allows to obtain a set of entities aiming to minimize the communication between them [13]. However, when using such approach, the resulting entities do not possess any associated semantics, and therefore it is not possible to analyze and modify them in a structured way.

Artificial intelligence community has proposed a modular decomposition of the mind, where a set of well defined entities cooperate as a society [14]. The concept of behavior, an entity encapsulating what is needed to warrant the accomplishment of a well-defined purpose, has been used to build complex intelligent systems. A behavior has a clear semantic in the context of the agent goals; which allow the creation of semantic relations between them, leading to form cooperative societies where rational conduct is obtained. The BESA architecture incorporates this behavior-oriented approach: an agent is composed of a set of concurrent behaviors.

2.2 Event-Driven

In the BESA model, an agent is seen as it is immersed in an environment populated of events. An event can be interpreted as a signal allowing to perceive that something interesting for an agent has occurred, and can include information about what has happened. What is really relevant is not the information, but the fact that the agent receives an stimulus and must react to produce a response. Notice that an event can be produced when something concerning the agent happens in the environment, including perhaps messages from other agents. Event in this case, the message itself is not the principal matter, but the fact that there is a communicative act with a well-defined intention.

In this event-driven approach, it is clear that an agent must be ready to react to multiple non-deterministic signals. A select mechanism, similar to the one defined in CSP, permits to deal with non-determinism providing a very powerful semantics that allows to implement concurrent programs in an elegant and clear fashion.

The abstract model of a select mechanism includes a set of guards. When the guard firing condition is verified, its associated treatment is executed. A firing condition is verified if both, a boolean expression is true and a communication event has occurred. Usually, the boolean expression is related to the state of the processing entity, and the communication event is associated to the reception of a predetermined type of message.

2.3 Social-Based

In order to analyze and design a MAS, the use of a social based model allows to study the system as an organization of interacting entities. Ferber has proposed a set of essential functions and dimensions to analyze an organization of agents; such approach has the advantage of identifying in a structured way the relations of the entities composing the system, as well as the connections with other systems [6].

In a social-based model an organization can be seen as it is composed of a set of low level organizations. This recursive decomposition of organization into simpler ones provides a framework to design MAS. At the higher level of abstraction the MAS is decomposed into a set of organizations with well defined semantics and functions, these organizations are broken again in a recursive fashion, finally entities assimilated to agents are obtained. The resulting agents have a clear semantic, which includes a goal in the organization and a well-defined interface with other agents and the environment.

At the lower level, the MAS is composed of a collection of plain agents; however, if a recursive decomposition has been accomplished, the model includes higher level abstract entities, organizations, which can be seen as agents serving as facilitators promoting the cooperation of low level entities. The BESA architecture aims to incorporate such social based approach. The idea is to provide elements to build patterns to manage and implement cooperation mechanisms and protocols between low level agents mediated by higher level ones.
3 BESA Architecture

The BESA architecture is composed of three levels: agent level, social level and system level.

3.1 Agent Level

The internal architecture of an agent integrates two important features: a modular composition of behaviors and an event selector mechanism. An agent is composed by a channel, a set of behaviors and a state. The state is a shared memory accessed by the agent behaviors using mutual exclusion synchronization. In figure 1 the agent components and their interconnection are presented.

The channel receives events addressed to the agent and implements a guard based selector scheme. A channel owns the unique entry point for events. In order to warrant that the semantics of the select mechanism is respected, all the events, even the internal ones, are received at the mailbox and must be processed by the guard selector. The guard selector transfers the incoming events to the appropriate port; the selection of the port is based on the event type. A guard firing asynchronous mechanism, which verifies if it can be fired, is activated when an event arrives or when a variable influencing the guard’s boolean condition is modified. When a guard is fired, the first event in the port is sent to the interested behaviors.

The behaviors are parallel processes working together to accomplish the agent goals. A behavior is intended to react to a well-defined set of event types; to show its interest in an specific event type, it binds itself to the associated guard, and an event will be sent to it when the guard is fired. Each behavior has a queue to receive events from the channel ports; when an event arrives, the behavior automatically executes the guard’s associated procedure. This procedure includes the appropriate program to react and produce a response to the received event.

A guard has two associated procedures. The first is executed automatically by the firing mechanism to verify the boolean condition. The second is executed by a behavior when the guard has fired.

One important feature of this scheme is that the code implementing a behavior is generic and can be written independently of the agent specificity. In fact, a behavior receives an event, and depending on its type executes a program. In consequence the agent designer only has to construct the internal structure of the agent, by binding behaviors, and to provide the procedures associated to each guard. Another feature is that agent mobility can be provided. In the code implementing a behavior, a synchronization can be included that suspends the behavior operation in a well-known point. When all the agent behaviors have been stopped, it can be safely moved. Once the agent context has been recovered, the behaviors resume execution at the predefined point.

3.2 Social Level

In the BESA abstract model, intermediate levels of abstraction are incorporated allowing to define and handle hierarchical organizations. An organization is defined as a set of agents associated to a mediator agent of the next higher level of the global organization hierarchy. The mediator acts as a facilitator of the cooperative work of the agents in the organization.

The mediator agent can serve as an event dispatcher receiving events and forwarding them to the concerned agents. It can also serve as a bridge to a pool of agents capable of rendering a service; in this case the mediator is registered in the yellow pages directory as the service provider, when a request event arrives to him, the event is addressed to the appropriate agent. In a more complex situation, the mediator can be designed to handle cooperative mechanisms, providing services for: task and resource allocation; conflict resolution; and distributed planning and coordination.

In this abstract social-based model, a mediator agent is seen as an entity that incorporates a set of associated agents. Nevertheless, when the system is deployed, it is not compatible with the agent concept to have agents inside agents. In fact, the system will be composed of a set of plain agents, where the links describing organizational relations between agents are only logical.
In BESA, the implementation of these social relations is supported by the agent internal select mechanism. A mediator service can be modeled and implemented by an interaction protocol, which is a predetermined sequence of interactions, event exchanges, between the involved agents.

An interaction can be modeled by a well defined event, which can be associated to a guard. Each agent involved in the protocol must include a behavior to handle the sequence of required interactions. For the designer of the system, the work is reduced to provide the associated procedures to a predefined set of guards; the semantic and data of these guards is determined by their associated interaction protocol.

### 3.3 System Level

An agent system implemented using the BESA architecture is seen as a distributed system composed of one or several BESA-containers, which can run in different physical/virtual machines. The whole system includes a unique FIPA compliant port, which receives communication and directory queries from other external systems using FIPA ACL. Figure 2 illustrates the components of a typical BESA system.
A BESA container is an execution space where agents inhabit; it manages their life cycle. A container has a local administrator, which supports agent communication and provides white and yellow pages directory services. The containers of a BESA system work together and implement a replication mechanism that makes transparent the agent naming, location and migration. When a new instance of an agent is created, the local administrator of the hosting container registers it in the white page directory. If required, an agent can register the services it provides in the yellow page directory. When a change in the life cycle of an agent is performed or any of the directories is modified, a replication mechanism is activated in order to maintain the consistency of all the local administrators.

The white page directory is composed by a collection of agent identification handlers. A handler include all the information needed to locate an agent in the system. When an agent requires to send an event to another one, it request to the local administrator the handler of the destination agent. If the agent is hosted in the local container, the address of the agent mailbox, where the event must be sent, can be obtained from the handler. If an external container hosts the agent, the information need to request the event delivery by the appropriated local administrator is provided.

4 Implementation

The architecture has been implemented using the Java programming language. This choice aims to run BESA systems in a heterogeneous, both hardware and software, platform.

The channel and the behaviors are implemented by concurrent threads, the run method, processing incoming events, is already coded and is not accessible to the agent designer. BESA provides a class hierarchy including basic abstract classes for agents, states, behaviors and guards. To implement an agent, derived classes must be created, and some few methods must be implemented. These methods include setup and setdown procedures for agents and behaviors, and the associated procedures for guard firing and execution.

At the startup, a container registers remote services in a predetermined RMI registry; then, a multicast message is sent in order to inform other already running containers about the existence of a new one. When a container receives this kind of messages, it updates its container table and registers itself in the new container. The new container synchronize its directories requesting this information from another container. Finally, the initial agents hosted by the container are instantiated.

In order to inform any change in the life cycle of an agent or in a directory entry, the local administrator invokes the replication services of the others administrators. This scheme allows to be fault tolerant, if a container crashes the system can continue. Nevertheless, in systems where agents are continuously created and removed, the replication mechanism can induce significant overhead.

The FIPA port is implemented as a thread receiving requests in a well-known socket port. It is seen as being part of one of the local administrators of the system. When a communication request arrives, its content is syntactically and semantically verified before it is transformed in a BESA event and posted to the appropriated agent. Directory services are provided to other external systems by the ACL port using the information available in the associated local administrator.

Conclusions

The agent and system levels of BESA has been implemented successfully. The social level will be constructed using the mechanisms provided by the other levels. Several applications are being developed, ranging from negotiation software agents to robotic physical agents. A multi-robot simulator has been implemented and will be extended to control a real robotic system.

An agent approach allows to solve complex problems in a natural way. BESA agents provide a framework to build highly concurrent and distributed applications, where the interacting modules have a clear and well-defined semantics, simplifying the system design and maintenance.
References

[1] Robocup Federation, “RoboCup Home Page”. 


http://www.fipa.org/resources/livesystems.


http://www.fipa.org/resources/livesystems.

