MULTIAGENT SYSTEMS AS SELF-MANAGED ARCHITECTURES IN SOFTWARE APPLICATIONS

Eva Cipi
Department of Informatics Engineering, University of Vlora, Albania
eva.cipi@yahoo.com

Betim Cico
Department of Informatics Engineering, Polytechnic University of Tirana, Albania
betim.cico@gmail.com

Abstract:
This work shows an architectural model of agent based system that can support us to assess the suitability of a multiagent system as a better solution to many technological problems. Agent internal architecture, the multiagent system organization, the communication infrastructure and other infrastructure services are some of included properties which we evaluate. The research presented here is a continuous work that develops a complex application integrating information agents in an architecture software engineering process. There are two main key aspects of this approach: modulated architecture software development and self-management. The functionality of the system can be achieved by collaborating subsystems. The centralized control is combined with the ability of agents to self-manage unexpected situations. We will revisit an agent based system which is developed by adding new functionalities such as autonomous supply chain services, control sales, with other properties we have designed in a market place.

Keywords: multiagent system, market place environment, robotic agent, self-management.

1. INTRODUCTION

Nowadays, software engineers have to expect several problems in solving/decision making processes and drive faster and easier to new ways in software developing area. The amount of information continues to increase and the list of requirements too. We need new strategies for working in engineering projects. If we know attributes of entities from the past works, we can preplan using entirely existing knowledge overlapping steps. Reusing past designs and learning, we gain time and avoid errors.

In the last work, we tried to use software agents as Database Management System components that allow a database systems may be configured and extended to support new requirements (Bass, Clements & Kazman, 2003). The software architecture was a hybrid agent based architecture used to be a subject of increasing efficiency in software engineering process.

This work shows an overview in designing process for multiagent systems in order to promote intelligent agents as modulated architectures for modelling simple rational behaviours in a wide range of software based applications. First, we will discuss about developing software engineering processes and techniques of multiagent system architecture decomposition in several levels. Then we give a case study of a subsystem designed step by step using the module decomposition in two levels. Starting from an application we have showed in the past, we are able to increase functionalities of management without considering a new system. The added application is a automated transportation service in a warehouse area. In light of these objectives, we have developed an application simulating a business environment. There are robot vehicles that work autonomously taking service point addresses from a wireless communication process with sales control agent which is part of a management system.

1.1. Architectural Design in the Development Life Cycle

The life-cycle consists of two main phases: developing the core system and delivering the final software product (Bandini, Manzoni & Simone, 2002).

In the first phase the core system is developed. This phase includes four activities. This phase includes four activities: defining a domain model, performing a system requirements analysis, designing the software architecture, and developing the core system. The software engineering process is an iterative process; the core system is developed incrementally, passing multiple times through the different stages of the development process.

In the second phase, subsequent versions of the system are developed until the final software product can be delivered. Assuming we have designed the core system (equal information system in the first work), we can develop other version with more functionalities.

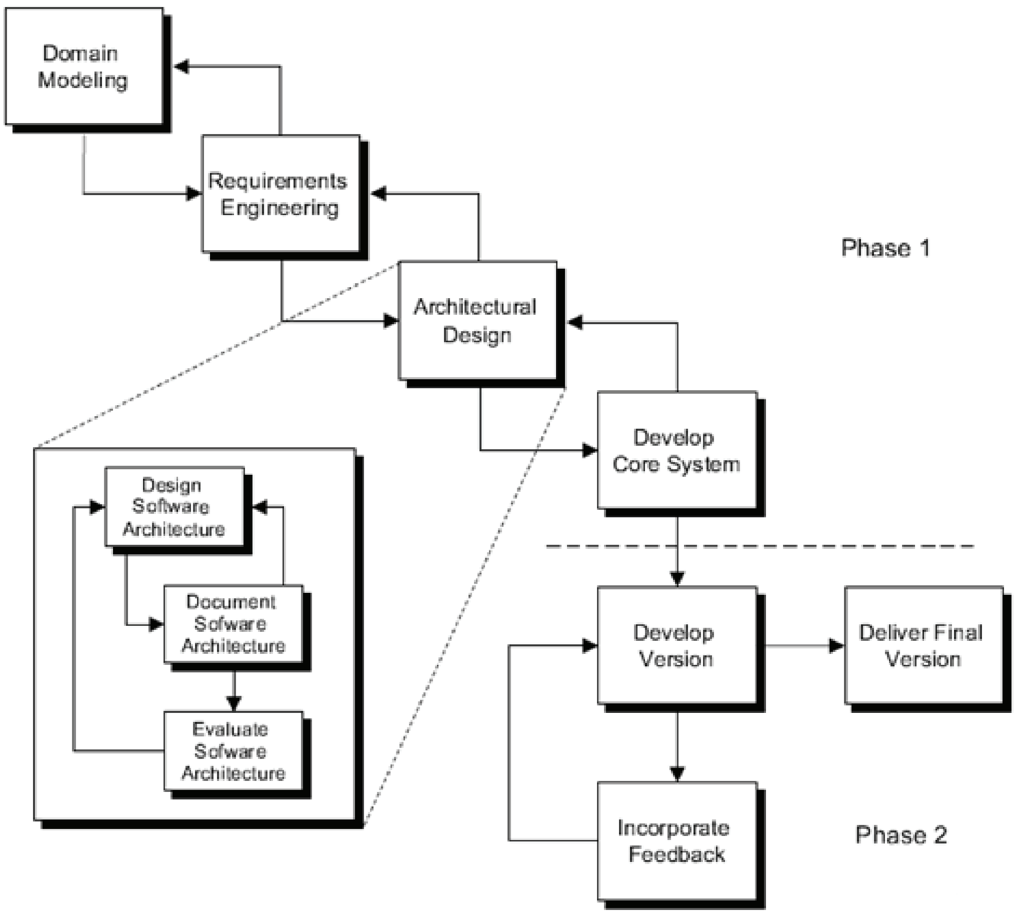
The architecture is important to software systems, because:

- Software architecture provides a basis for creating communication, mutual understanding and consensus about the software system among stakeholders.
- Significant influence on system qualities and
- Software architecture is a reused and transferable abstraction for other system with similar requirements. (Buchmann & Bass, 2001)

1.2. Multiagent Systems and Software Architectures

A multiagent system provides the software to solve a problem by structuring the system into a number of interacting autonomous entities embedded in an environment in order to achieve the functional and quality requirements of the system. In particular, a multiagent system structures the system as a number of interacting elements in order to achieve the requirements of the system. This is exactly what software architecture is about. Clements, Kazman & Klein (2002) define software architecture as:

Picture 1: Architectural Design in Software Development Life Cycle



Software elements (or in general architectural elements) provide the functionality of the system, while the required quality attributes (performance, usability, modifiability, etc.) are primarily achieved through the structures of the software architecture.

Typical architectural elements of a multiagent system software architecture are agents, environment, resources, services, etc. The relationships between the elements are very diverse. In short, multiagent systems are a rich family of architectural approaches with specific characteristics, useful for a diversity of challenging application domains (Steegmans, Weyns, Holvoet & Berbers, 2004).

There are applications that provide different levels of complexity and various forms of dynamism and change. The architecture for many of them is a set of agents, for reuse, they can serve for developing software architectures transporting them as ready entities with specific attributes such as: each agent has incomplete information or capabilities for solving the problem and, thus, has a limited viewpoint and computation is asynchronous.

A multiagent system consists of a (distributed) environment populated with a set of agents that cooperate to solve a complex problem in a decentralized way (Weyns & Holvoet, 2006). Behavior-based action selection is driven by stimuli perceived in the environment as well as internal stimuli. Situated agents employ internal state for decision making relates to:

- planning off line (static information of the system);
- planning on line (dynamic information related to the changes of the environment); or issues internal to the agent.
- environment encapsulates resources and enables agents to access the resources (Weyns, Vizzari & Holvoet, 2005).

1.3. Self-Management in Software Applications

The general idea of self-management is to endow computing systems with the ability to manage themselves according to high-level objectives specified by humans. Researchers divide self-management into four functional areas (Zambonelli & Parunak, 2002):

- Self-configuration: automatically configure components to adapt themselves to different environments;
- Self-healing: automatically discover, diagnose, and correct faults;
- Self-optimization: automatically monitor and adapt resources to ensure optimal functioning regarding the defined requirements; and
- Self-protection: identify and protect against attacks.

The environment will occupy an important role. An agent based system which pretends to be self-managed must take the appropriate actions based on a sensed situation in the environment. This requires functionality for monitoring, decision making, and action execution. This requires coordination of behaviour of agents used in different applications.

2. APPLICATION: AN AUTOMATED SYSTEM IN A MARKET PLACE

In this research we propose a modulated architecture agent based approach to develop such complex applications. The approach aims for extending the functionalities of systems that are able to manage dynamism and changes in their environment autonomously.

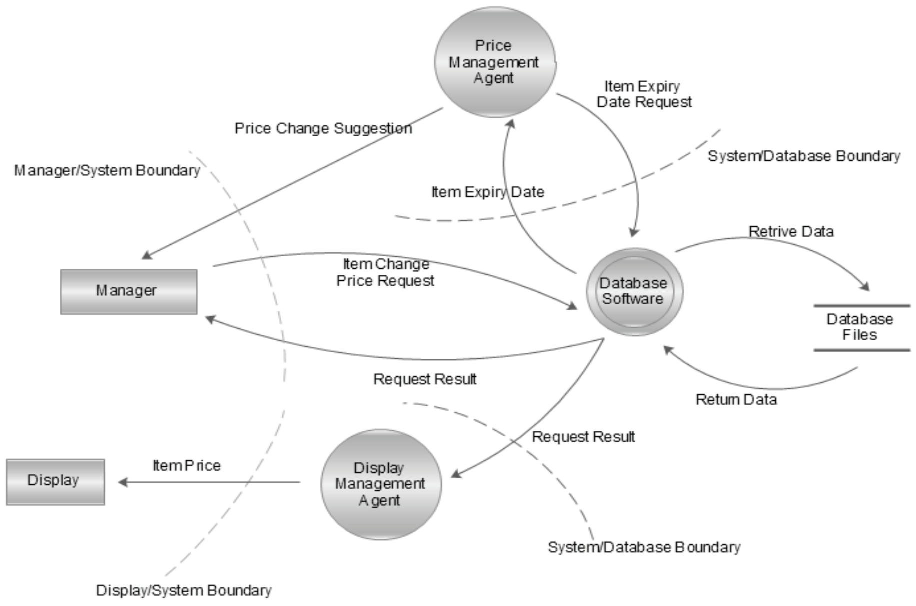
2.1. Related Work

In the last work we showed an application based on simulating an information management system which uses information agents well defined to act and to do specific actions in a market place environment. We tented in two objectives:

- Agent based systems can offer the needed tools for expertise storing in a database management system
- The modularity of the architecture allows us to add other agents without changes (Cipi & Cico, 2010)

The system is based on database files which take all the information. The agency includes several services made by agents: expertise of selling and inventory (selling agent), display the changes of prizes (display agent), expertise order amounts (order agent) and suggestions of prices (price agent).

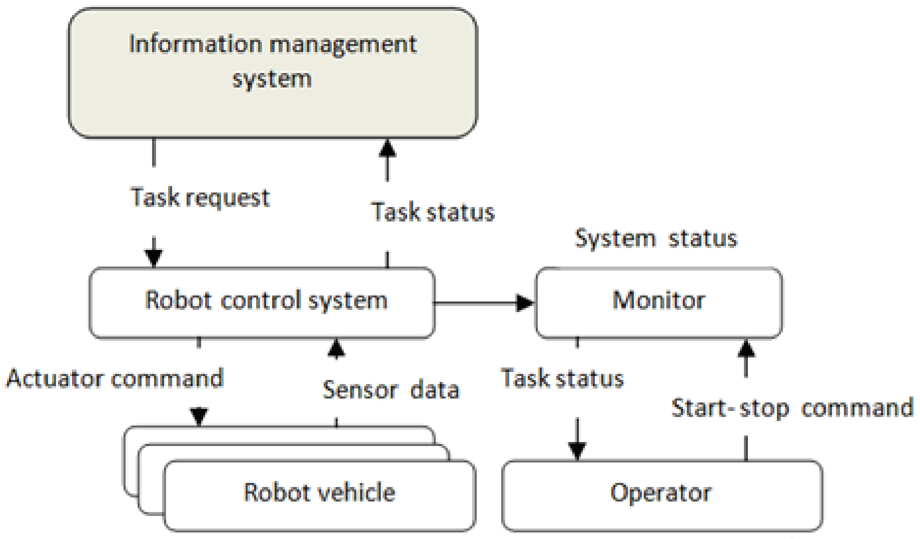
Picture 2: Agent Data Flow Diagram into Management System



2.2. Continuous Work

Here we show new functionalities of system by adding automatic service transportation. The robot has to transport loads from one place to another. The tasks are generated by the information agent which is part of central information systems; typically for a business management program (we have discussed it in the past work).

Picture 3: Diagram of the robotic system for semi-automatic transportation



The task is composed out of some processes like receiving order from sales control agent acquiring the first service point address, moving to first service point, receiving the second service point, moving to the second service point. In order to execute this task, the system has to perform:

1. Route assignment: paths are generated by the information systems and have to be assigned to robot vehicle that can execute them.
2. Routing: the robot must route efficiently through the environment layout of the warehouse when executing transports.
3. Gathering traffic information: although the layout of the system is static, the best route for the robot in general is dynamic, and depends on the actual traffic condition. Using visual sensors, the robot routes efficiently without collision with other objects (Kephart & Chess, 2004).

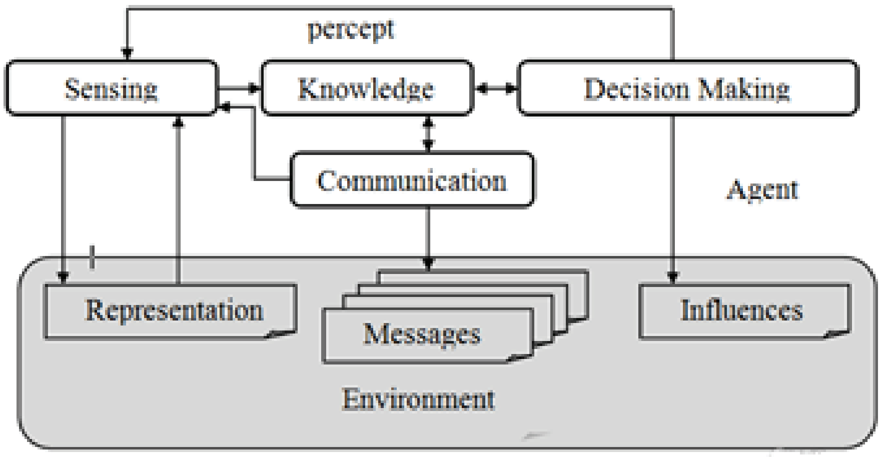
2.3. The architecture

This model integrates the environment and agent integrating mechanisms of adaption for agents. We have divided the model in two parts: environment and situated agent.

Environment Model

The environment model consists of a set of modules with flows between the modules. The modules represent the core functionalities of the environment. The model consists of two main modules:

Picture 4: Decomposition of agent in four modules



- the deployment context (referred to the given hardware and software and external resources with which the multiagent system interacts (sensors and actuators, a printer, a network, a database, a web service, etc.).and
- the application environment.(refereed to the part of the environment that has to be designed for an application).

Agent Model

The agent model consists of four modules with flows between the modules. Figure 3 shows the model. The modules represent the core functionalities of a robotic agent. Knowledge module provides the functionality to access and update the agent's current knowledge. Sensing module receives perception requests from the Decision Making and Communication modules to update the agent's knowledge about the environment. Decision making and

communication use the agent's current knowledge to make appropriate decisions. Communication module writes the location in the agent's current knowledge which in turn will be used by the decision making module to move the agent efficiently towards the destination. Decision Making provides the functionality to an agent for selecting and invoking influences in the environment. Decision making consists of two basic functions:
influence selection and actuator execution. To select appropriate influences, an agent uses a behaviour based action selection mechanism extended with roles and situated commitments. Execution provides the functionality to invoke selected influences in the environment.

Picture 4: An overview of a robotic vehicle application

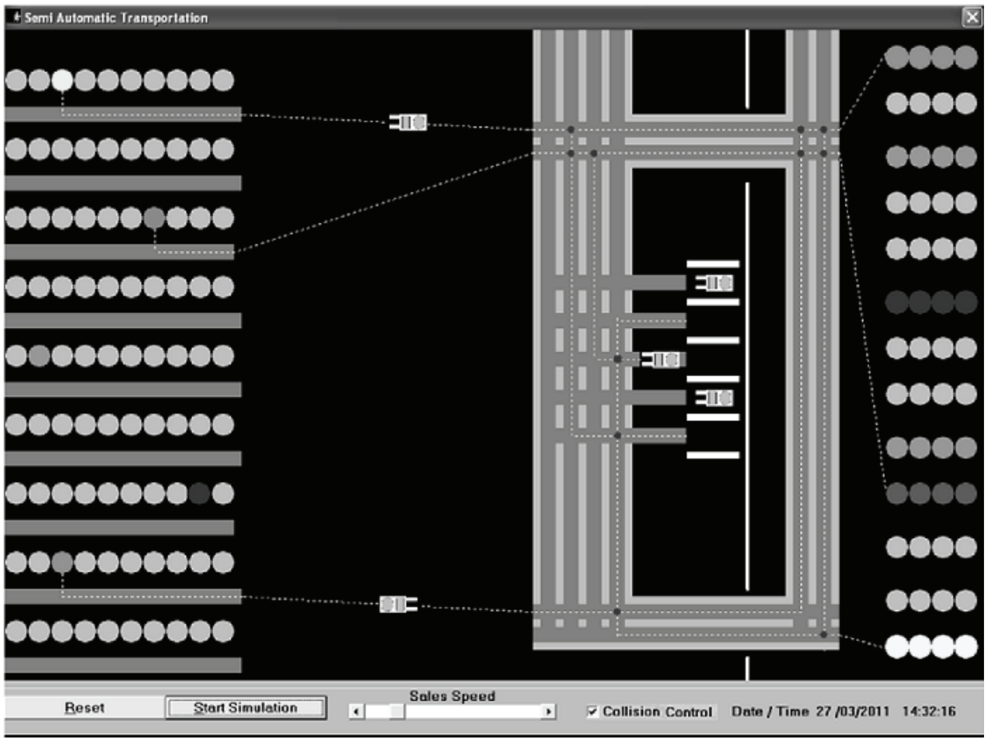

Semi Automatic Transportation
Reset
Start Simulation
Sales Speed
Collision Control
Date / Time 27 /03/2011 14:32:16

Figure 4 shows the simulation interface where there are five robotic vehicle agents. We consider them independent entities that communicate with a management system which generates transportation tasks through a sales control agent.

The colored circles shows service points which are generated from the management system that consists in the control of task distribution. Path calculation of robotic vehicle is preplanned. If there are collision points (red points), agents percept and calculates autonomously a new path.

3. CONTRIBUTIONS

This research contributes with a new perspective on software engineering for multiagent systems. Specially, this research integrates multiagent systems as software architecture in a general software engineering process. Concrete contributions are:

- A developed model for multiagent systems that links in particular the model with two entities: the environment to a first-level abstraction that can be used in the design of situated multiagent system applications and extends models of agents to

an integral model that includes support for different concerns, such as perception, behaviour, and direct communication.
- A modulated architecture that maps the advanced model for multiagent systems onto a system decomposition
- This architecture for multiagent systems integrates a set of architectural best practices from various applications we have studied and built.
- The architecture also offers a robotic vehicle as a case study to learn the advanced perspectives on software developing process.

The application of a multiagent system in a complex information management system that uses automatic vehicles to transport products satisfy a set of requirements such as functional requirements of the system, quality requirements, incremental development of a software architecture, evaluation of the software architecture, incremental implementation of the application, testing to verify the main system requirements.

REFERENCE LIST

1. Bass, L., Clements, P. & Kazman, R. (2003). Software Architecture in Practice. *Addison Wesley Publishing Comp*, 95–97.
2. Bandini, S., Manzoni, S. & Simone, C. (2002). Dealing with Space in Multiagent Systems: A Model for Situated Multiagent Systems. *In 1st International Joint Conference on Autonomous Agents and Multiagent Systems.* ACM Press.
3. Buchmann, F. & Bass, L. (2001). Introduction to the Attribute Driven Design Method. *23rd International Conference on Software Engineering, IEEE Computer Society.* Toronto, Ontario, Canada.
4. Clements, P., Kazman, R. & Klein, M. (2002). Evaluating Software Architectures: Methods and Case Studies. Addison Wesley Publishing Comp.
5. Steegmans, E., Weyns, D., Holvoet, T.,
6. & Berbers, Y. (2004). A Design Process for Adaptive Behavior of Situated Agents. In Agent-Oriented Software Engineering V, 5th International Workshop, AOSE, New York, NY, USA, Lecture Notes. *Computer Science*, *3382*.
7. Weyns, D. & Holvoet, T. (2006). Multiagent systems and Software Architecture. *In Special Track on Multiagent Systems and Software Architecture, Net.ObjectDays, Erfurt*, Germany.
8. Weyns, D., Vizzari, G. & Holvoet, T. (2005). Environments for situated multiagent systems: Beyond Infrastructure. *Proceedings of the Second International Workshop on Environments for Multi-Agent Systems, Utrecht, Springer Verlag.,* Lecture Notes. *Computer Science*, *3380*.
9. Zambonelli, F. & Parunak, H. V. D. (2002). From Design to Intention: Signs of a Revolution. *1st International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Bologna, Italy, ACM Press, New York.
10. Cipi, E. & Cico, B. (2010). Information Agents as a New Paradigm for Developing Software, Applications in Database Systems, Conference Proceeding DSC 2010, Thessaloniki, Greece, *1*, 514–519.
11. Kephart, J. & Chess, D. (2004). The Vision of Autonomic Computing. *IEEE Computer Magazine*, *36*(1).