Agent-based Software Engineering as a Layered Technology

Mohammad Reza Ayatollahzadeh Shirazi  
Intelligent Systems Laboratory  
Computer Engineering Faculty  
Amirkabir University of Technology  
Tehran/Iran  
ashirazi@ce.aku.ac.ir

Ahmad Abdollahzadeh Barfournoush  
Intelligent Systems Laboratory  
Computer Engineering Faculty  
Amirkabir University of Technology  
Tehran/Iran  
ahmad@ce.aku.ac.ir

Abstract

In this paper, a new approach to agent-based software engineering is presented. According to this approach, agent-based software engineering is a layered technology that encompasses software quality, agent-oriented software development process, agent-oriented methods and agent development tools. This pragmatic approach guides developers and project managers to remove many pitfalls of agent-based software engineering, helps researchers to determine the existing gaps in the current works and helps tool developers to organize their efforts for constructing agent-oriented tools. On the basis of this approach an investigation of the current works in agent-based software engineering is presented. Also, in this paper, umbrella activities for complementing the main activities and steps in the agent-based development are introduced and described.

1. Introduction

Now an increasing number of problems in industrial, commercial, medical, networking and educational application domains are being solved by agent-based solutions. These software solutions are mainly complex, open and distributed. The key abstraction in these systems is the agent. An “agent” is an autonomous system that interacts with its environment in order to satisfy its design agenda. This concept is a natural metaphor similar to “object” for software engineers and they can understand, model and implement many systems on the basis of an autonomous and interacting agents. Agent-based software engineering is a powerful way of approaching large scale software engineering problems and developing agent-based systems. In this approach to software development, applications are written as agents.

This paper presents a practitioner’s view to agent technology and focuses on the engineering aspects of agent-systems. In this paper, we introduce and justify agent-based software engineering as a layered technology. The layers of this technology are quality, agent-oriented software development process, agent-oriented methods and agent development tools. This approach guides developers to remove many pitfalls of software engineering with agents, helps researchers to recognize the existing gaps in the current works and helps tool developers to organize their efforts for constructing agent-oriented tools. Also, we introduce and describe umbrella activities in agent-based software engineering.

The reminder of this paper focuses on our insight into what is our approach to agent-based software engineering. Section 2 reviews agent-based software engineering. Section 3 presents and describes our new approach to agent-based software engineering and investigates the current works in this field. Section 4 discusses our proposed activities for complementing the main activities in agent-based software engineering. Finally, we present our conclusion.

2. Agent-based Software Engineering

Agents are a new paradigm for developing software applications. Researchers have considered agent-based computing as ‘the next significant break-through in software development’, and ‘the new revolution in software’. We have found that most researchers have an agreement on the following definition for an agent [3]:

“An agent is a computational system situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives.”

[39,42]

After many years of research, we think that most researchers have converged to a set of properties for an agent. The most agreed properties of agents are autonomy, situatedness, reactive, sociable and pro-active [3]. An agent-based system is a system which the key abstraction used is that of an agent [22]. Multi-agent systems are agent-based systems that are designed and implemented as several interacting agents and are both more general and significantly more complex than the single-agent system. Therefore, an agent-based system is designed and implemented in terms of agents. Agent-based applications have common characteristics such as complexity, openness and distribution of data or control. Also, many of them are real-time and safety-critical systems. So, they should be reliable and high quality systems. We can say that development, operation and support of agent-based systems need application of systematic, disciplined and quantifiable approaches. It is necessary to develop software engineering techniques that are specially tailored to agents. So, we need agent-based software engineering. Agent-based software engineering
is software engineering for agent-based systems. Agent-based software engineering represents a novel and powerful way of approaching large scale software engineering problems [33,41]. In this approach to software development, applications are written as software agents. We consider agent-based software engineering as a layered technology. In the next section, we describe our approach to this discipline.

3. Layers of Agent-based Software Engineering

Software engineering is a layered technology [30]. In this view, software engineering encompasses a process, technical methods and tools. According to this view, we approach to the agent-based software engineering as a layered technology. As shown in Figure 1, this technology encompasses agent-oriented software development process, agent-oriented methods and agent development tools. Of course, the basis of agent-based software engineering is quality. So, for developing high quality and efficient agent-based systems in a timely and cost-effective manner, it is important to have software quality control and assurance techniques, agent-oriented software development process, agent-oriented analysis and design methods, agent construction methods, agent languages, testing methods, support methods and tools for supporting agent-oriented processes and methods.

According to this approach, the researchers in the field of agent-based software engineering can organize the existing works, determine gaps in the current work and focus on the needed further works. Also, it is a good framework for study the pragmatic aspects of agent system development. We claim that if we have this approach to agent-based software engineering, many pitfalls that await the agent system development projects [43] will be removed.

<table>
<thead>
<tr>
<th>Umbrella Activities in Agent-Based Software Engineering</th>
</tr>
</thead>
<tbody>
<tr>
<td>CM-AbS</td>
</tr>
<tr>
<td>Agent Development Tools</td>
</tr>
<tr>
<td>RM-AbS</td>
</tr>
<tr>
<td>Agent-oriented Methods</td>
</tr>
<tr>
<td>SQA-AbS</td>
</tr>
<tr>
<td>Agent-oriented Software Development Process</td>
</tr>
<tr>
<td>SM-AbS</td>
</tr>
<tr>
<td>Software Quality</td>
</tr>
</tbody>
</table>

Figure 1. Four layers of agent-based software engineering technology

Because there is a fundamental mismatch between the concepts used by existing software engineering methods and techniques and the agent-oriented view, many existing approaches fail to adequately capture an agent’s flexible, autonomous problem solving behavior, the agent’s interaction, and the complexity of an agent’s system’s organizational structure [40]. Although we can use and extend existing works in software engineering, we need novel and new techniques and methods in the area of agent-based software engineering. In the following sections, we mention some representative samples of existing works in each layer.

3.1. Agent-oriented Development Process

Agent-oriented development process is a set of steps for producing a timely and high-quality agent-based system. MASSIVE [25] is one example of an agent-oriented software process model for development of multi-agent systems. This process model is an iterative view engineering approach that is itself based on iterative enhancement and round-trip engineering in the special context of multi-agent systems. Tropos [7] is another sample of an agent development process. Tropos is intended to support five phases of software development: Early requirements, Late requirements, Architectural design, Detailed design and Implementation, where the actual implementation of the system is carried out in JACEK [2] consistently with the detailed design. Park et al. [28] propose a linear-sequential process model that contains domain analysis, agent modeling, agent design, agent implementation, agent integration and verification and validation activities. Some agent frameworks have their own agent development process. For example, ZEUS has a linear-sequential model that consists of Domain analysis, Design, Realization and Runtime support activities [26].

Wooldridge and Jennings have identified some of the main pitfalls of agent system developments [43]. One of these pitfalls (Pitfall 4.3) is that “You forget you are developing software”. The layered approach to agent-based software engineering helps developers to overcome this problem. The agent development process helps project managers and developers to remember that they are developing software.

3.2. Agent-oriented Methods and Tools

In order to produce any kind of software, we need methods. These methods tell the software engineer how to build software. Agent-based systems are complex software and building these types of software needs agent-oriented methods. These methods encompass tasks that include requirement analysis, design, agent construction, agent-oriented testing and support.

3.2.1. Agent-oriented Analysis and Design

There are some agent-oriented methods for analysis and design of agent-systems. Each of these introduces a process for the analysis and design of agent-based systems, a set of diagrams and notation for documenting the artifacts of process and creating analysis and design
models. These methods allow an analyst and designer to go systematically from a statement of requirements to a detailed design. Some representative samples of these methods are Gaia [40], MaSE [10,37], MESSAGE/UML [9], Agent-Oriented Analysis and Design [8], Agent Modeling Technique for Systems of BDI Agents [18], MAS-CommonKADS [19] and CoMoMAS [18]. These methods are extensions or further abstraction of the existing object-oriented and knowledge engineering methods. All these approaches define some analysis and design models for an agent system. These models try to identify and capture roles and their responsibilities and goals (e.g. Role Model in Gaia and MaSE or Organizational Model in AAI Methodology), capture and represent dependencies and relationships between the various roles (e.g. Interaction Model in Gaia), develop an organizational structure and elaborate the knowledge and behaviors associated with a role or agent (e.g. Services Model in Gaia).

There are some design methods for agent-based systems [33]. Yim et al. [36] suggest an architecture-centric design method for multi-agent systems. The method is based on standard extensions of UML. Bergenti and Poggi [6] try to exploit UML [27] in design of multi-agent systems at agent level. They propose a set of agent-oriented diagrams for creating UML-based models for multi-agent system architecture, agent ontology and agent interaction protocols. Aridor and Lange [4] suggest a classification scheme for design patterns in a mobile agent context. Kendall et al. [23] suggest a seven-layer architecture pattern for agents, and sets of patterns belonging to each of the layers. Erol, Lang and Levy [11] suggest a three-tier architecture that enables composition of agents by applying reusable components. Also, formal methods are used in specification of an agent system. One example of such methods represents an agent-based system by using mental states of agents such as beliefs, desires, and intentions [41].

3.2.2. Agent Languages

An agent language is a system that allows one to program computer systems in terms of agent theory concepts [42]. In this language some attributes of agency (beliefs, goals, or other mentalistic notions) are used to program agents. One of the major contributions in developing agent languages is Shoham’s proposal for agent-oriented programming (AOP) [32]. In this ‘new programming paradigm’ the key idea is directly programming agents in terms of the mentalistic, intentional notions. Shoham introduced an agent programming language (AGENT-0) for writing agent programs. He theorized that cognitive agents possess a mental state which is composed of various mental elements: beliefs, commitments, capabilities, and commitment rules. Thomas extended the AGENT-0 research and developed PLanning Communicating Agents (PLACA) [34]. PLACA is an agent programming language similar to AGENT-0 with extensions for planning. Concurrent METATEM [14] is another example of agent languages.

3.2.3. Agent-oriented Testing

The high abstraction level of agent programming and the complexity of agent-based systems produce a need for methods and tools to analyze system behavior. So, we need testing and debugging methods on both code and agent level. This includes visualization of the organizational structure, development of the agent society during runtime and the visualization of agent interaction on “speech act level” [31] to ensure that the interactions between agents are consistent with the goals of the system. Of course, there is a need for further developing standard testing methods for agent-based systems and it is another area of further work for the agent researchers.

A new work in this area is formal verification of agent communication protocols [10]. The main idea is before a multi-agent system can be trusted to perform as expected, the communication protocols between the agents must be formally verified. In this work a formal methodology that automatically verifies the interaction between agents is introduced. The verification process includes checking that behavioral models of interacting agents respond to agent messages correctly. An agent conversation analysis technique is proposed in [15]. In this technique that is supported by the JIVE [15] framework, the conversations and conversation rules of the multi-agent system are represented by using a finite state machine. This representation allows for Petri-Net description of the conversations. By using this representation the design is validated.

Another layer in agent-based software engineering is agent-based development tools. These tools provide support for analysis, design, construction, test and maintenance of agent-based systems. In the past years a number of commercial and academic products have emerged to support agent-based development process and methods [15,21,26,37]. According to our investigation and surveys, we classify the existing agent tools in these categories: agent-oriented analysis and design, agent development frameworks, testing and debugging, agent prototyping. A discussion on these tools and technical aspects of them can be found in [1].

4. Umbrella Activities in Agent-based Software Engineering

In agent-based software engineering, we need other activities for complementing the main activities and steps in the agent-based development. We propose the following activities for applying throughout the software process. For each activity, the need for it and areas of further works in this field are described.

Software Measurement for Agent-based Systems (SM-AbS): In measurement process for agent-based systems, we should define some metrics that are appropriate for some representation of agent-based system. These metrics provide an indication of the system quality. These metrics can be defined for agent-oriented analysis and design models, agent programs, agent-oriented testing and agent-
based system support. For example, designer of a multi-agent system can develop a solution with few agents and bundle all the functionality in these agents. This solution can not satisfy design goals of agent-based systems and fails to exploit the power of these systems. Here, we can define coherence metric as a design metric for the agent-based system and measure the coherency of the designed system by using this metric.

Also, we should define measures for the quality of the produced agent-based system. Some good measures are correctness, maintainability, integrity, and usability. As a starting point for defining quality measures for agent-based systems, some attributes for agent architectural evaluation has been defined at the SEI [38]. These quality attributes are performance, predictability, security, adaptability, availability, and fault tolerance. After defining appropriate measures and metrics, data collection for the metrics and analysis of these metrics are the next activities in the measurement process. A good measurement process for an agent-based system removes many pitfalls in agent-development projects and leads to a higher quality system.

Software Quality Assurance for Agent-based Systems (SQA-AbS): These activities are needed to ensure the conformance of the agent-based system to the stated functional and non-functional requirements and development standards. Software safety is one of these activities that focuses on the identification and assessment of potential hazards. When an agent-based system is used in a safety critical domain doing this activity becomes more important. We need techniques for analyzing and evaluating faults in the context of an agent-based system.

Configuration Management for Agent-based Systems (CM-AbS): It is a set of activities for managing change through the life cycle of an agent-based system. These changes are due to new user requirements, changing project strategy or removing bugs. We believe that these changes can be done at agent level and code level. At agent level, the engineer makes changes in agent society and its organizational structure, agent mental states, system ontology, coordination and cooperation protocols, agent architecture or agent communication language. At code level these changes are due to modifications and corrections in the agent(s) program code. We should perform identification of configuration objects, version controlling, change controlling, configuration auditing, and reporting for the agent-based systems.

Risk Management for Agent-based Systems (RM-AbS): Agent technology is a new and leading-edge technology. Sometimes the purpose for the use of this technology and degree of applicability of it are not clear for managers and developers. This technology is not a right solution to every problem. So, there is a technical risk in projects for using the agent technology as an appropriate solution. Managers and developers should analyze the degree of applicability and suitability of this technology for using in their projects. Even after choosing this technology, the agent-based project will have project risks, technical risks, and business risks. All these risks should be managed in agent-based projects. Applying agent technology for solving all problems initiates agent projects with no clear goals, reasons, and criteria for assessing the success or failure of the project. Forgetting agent-oriented process, methods, and tools or choosing inappropriate process, methods, tools, and development strategies are examples of risks in agent-based projects.

Project Management for Agent-based Systems (PM-AbS): Project management focuses on the four P's: people, product, process and project [30]. In agent-based software engineering we need to new roles such as agent architect, agent developer, and agent system engineer for people. Techniques for software scoping and problem decomposition should be developed. The project manager should consider the agent-oriented process or any other software process that is appropriate for developing the agent-based system. The project manager should have some quantitative measures to assess the status of an agent-based project and track the potential risks.

As mentioned earlier, in some cases we can use existing software engineering techniques and activities or we can adopt them for agent-based systems, but we think that still there is a need for new agent-based techniques and methods for these activities.

5. Conclusion

This paper presented a practitioner's view to agent-based software engineering. We talked about necessary and important practices in this field. Our important aim in this paper was to introduce the main new concepts and techniques that we need in the agent-based software engineering and area of further works in this field. So, a new approach to this discipline presented and it is believed that this approach can guide developers to remove many pitfalls of software engineering with agents, help researchers to recognize the existing gaps in the current works and help tool developers to organize their efforts for constructing agent-oriented tools. Also, we introduced the umbrella activities in agent-based software engineering. These activities were concerned with quality control and assurance, measurement, configuration management, risk management, and project management of agent-based systems that are applied through the development process of these systems. We believe that for producing a high-quality agent-based system, we need to consider these activities in the agent-based projects.

6. References


