A Review of Agent-Oriented Development Methodologies and Programming Languages/Frameworks

Khalil Salah *salah.kh@gmail.com*
Advanced Informatics School
Universiti Teknologi Malaysia
Kuala Lumpur, 54100, Malaysia

Ardavan Ashabi *ardavan.ashabi@gmail.com*
Advanced Informatics School
Universiti Teknologi Malaysia
Kuala Lumpur, 54100, Malaysia

Abstract

Agents are software systems and can be associated with an entity, framework, architecture and even languages. They are piece of program codes that are able to autonomously complete tasks. Before developing an agent, methodology to be used in development should be clarified and based on the methodology, suitable programming language and framework should be selected. This paper reviews three agent development methodologies (Prometheus, Tropos, and MaSE) and six agent-oriented programming languages/frameworks (MetateM, IndiGolog, Brahms, GOAL, JIAC, and Agent Factory).

Keywords: Computer Agents, Agent-Oriented Programming, Agent-Oriented Methodologies.

1. INTRODUCTION

Agents are entities with mental modules. They can have beliefs and capabilities also they are able to choose, and commit [1]. They are piece of program codes that are able to autonomously complete tasks. Autonomy is one of the properties that each agent has. It means the agent shall operate without direct human interference. Moreover, agents should be able to interact with each other or with humans. This characteristic is the social ability of an agent. Reactivity and pro-activeness are also two important property of each agent. Reactivity is to feel the environment and respond to environment changes in timely manner; and pro-activeness means exhibiting goal-directed behavior [2, 3].

Instantiated from Object Oriented Programming (OOP) paradigm, Agent-Oriented Programming (AOP) views the system as modules that communicate with each other by handling incoming-outgoing messages and by fixing the mental state of agents to consist components such as beliefs, capabilities, and decisions, AOP make a special framework. There are different limitations on the mental condition of an agent and they approximately match to constraints on their intellectual state [1]. When implementing agents as a system, they have to be capable of achieving sophisticated goals autonomously until they find a solution to complete the goal [2]. Capabilities of systems based on agents are large and it is a potential to exploit this technology. The successful future of agent-based systems depends on a complete and accurate analysis of its conceptual basis, structuring principles, and methods used for the design and development of agent systems. Therefore, having a methodology to support changes in requirements and new components is highly encouraged [4].

This paper reviews some methodologies and programming languages/frameworks which are used in developing agents.

2. DEVELOPING AGENTS

Before developing an agent we need to understand the concepts, current methodologies and programming languages of agents. As mentioned earlier, agents are software systems which can be associated with an entity, framework, architecture and languages. They are programs able to autonomously complete assigned tasks. They may be required to adapt, learn or collaborate with stimuli, sensors and actuators or data flows [2]. Methodologies are needed in all aspects of software development. In [5] the author mentions that having a methodology and following it while developing a software system, is a key factor for successful development process. Especially when software applications are complex, and difficult to develop a well-defined development process will reduce the risk of failure. In the same way, while developing agents, regarding suitable methodology should be selected regarding the type, size and purpose of the development.

2.1 Methodology

Regardless of advances in agent-oriented methods and technologies, due to lack of systematic methodologies to guide the development process, they have a limited rate of adoption in large-scale systems. So, agent-oriented methodologies play a significant role in this area. Current accepted methodologies provide different modelling concepts and analysis techniques together with some part for supporting tools. Maturity and scope of coverage are different in these methodologies [6]. Below three famous methodologies are reviewed:

Prometheus

Prometheus methodology is an Agent-Oriented Software Engineering methodology (AOSE) with focus on the development of intelligent agents rather than black boxes [7]. It has three phases of: (1) System Specification, (2) Architectural Design, and (3) Detailed Design [3]. All software engineering activities are included in Prometheus, from requirements specification to detailed design and implementation [7]. Each phase has several models with focus on dynamics of the system and whole system structure or structure of system components. Textual descriptor forms that provide the details for individual entities are also included in each phase [3].

During system specification, functionalities are defined and by using goals as scenarios the system will be specified; its interface with specified environment is described in terms of actions, percepts and external data. In architectural design, the types of agent will be identified and the system structure will be captured. In detailed design, internal development of each agent will be developed and defined in detail [3].

Tropos

Tropos is a methodology with focus on analysis of requirements at its early stage. It supports all software development process activities, from system analysis to the system design and system implementation [4]. This methodology provides a general interface to different development process activities, documentation and evolution of the software rather than an incremental refined and extended model. This analysis process allows the reason for developing the software to be captured. Development process in Tropos consists of five phases: (1) Early Requirements, (2) Late Requirements, (3) Architectural Design, (4) Detailed Design and (5) Implementation [4]. Requirements analysis in Tropos is split in two main phases of Early Requirements and Late Requirements analysis. In the Architectural Design and the Detailed Design phases the focus is on the system specification, regarding the requirements results from phase 1 and phase 2. And finally in Implementation the detailed design specification will be followed [4].

MaSE

MaSE is Multi-agent Systems Engineering that provides a methodology covering all aspects of systems' lifecycle and assists developers in designing and developing multi-agent systems. It is a large-scale methodology being used in the analysis of multi-agent systems. In MaSE a strong framework for design and development of multi-agent systems is provided by taking advantages of goal-driven development, and using the power of multi-agent systems in defining roles, protocols and tasks. Another advantage of this methodology is that the steps are defined one by one which results a simple transition between models [8]. In MaSE the processes are described from an initial system specification to system implementation which will provide a road map for the system developer. These process consist of seven steps, which are grouped into two phases [9]: (1) *Analysis phase* which consists of three steps: (1-1) *Capturing Goals,* (1-2) *Applying Use Cases*, and (1-3) *Refining Roles*. The (2) *Design phase* has four steps: (2-1) *Creating Agent Classes,* (2-2)*Constructing Conversations,* (2-3) *Assembling Agent Classes*, and (2-4) *System Design* [8].

2.2 Agent Programming Languages/Frameworks

Agents are entities with intelligent behavior and developing them needs to satisfy their basic characteristics. They should be able to think and have relationship between each other. To satisfy these required characteristics, "researchers have drawn heavily on formal models of agents and on agent logics, including epistemic logics, logics of action, dynamic logic, coalition logics, etc." [10]. Below some of the agent programming languages/frameworks are reviewed.

MetateM

MetateM is an agent-oriented programming language which is developed as part of research into formal methods for developing software systems[11]. It is strongly based on the theory of temporal logic and methods of formal software development are used in it. MetateM is using temporal logic as a tool to specify and model reactive systems by directly applying the logic and meta level reasoning, since identifying meta language and language as one and the same, provides a highly reflective system [12]. MetateM can be grouped in declarative languages, which allows describing the behavior of the agents by logical rules and Meta statements. It is the only specification language that is able to be executed directly which based on agent's specification, will ensure the execution of a multi-agent system is correct.

IndiGolog

The IndiGolog agent-oriented programming language targets agents which sense the environment and plan for their goals. A high-level execution is supported in IndiGolog. The developer develops a high-level non-deterministic program with domain specific actions and the IndiGolog thinks about the conditions and outcomes of the actions in the program to find a legal terminating execution. To achieve this, the developer declares the specifications of the domain (primitive actions, preconditions and effects, what is known about the initial state) in the situation calculus. The amount of non-determinism in the program can be controlled by the developer and IndiGolog supports concurrent programming [13].

Programs in IndiGolog can be intelligent enough to get required information during execution and react to external actions. The language can be used in development of robot control applications that combine planning, thinking, and reactivity. Complex agents are supported in IndiGolog which are able to think, plan, react, monitor execution and sense the environment. Specification of domain dynamics and specification of behaviors are included in IndiGolog agent. Because implementation of IndiGolog is in Prolog, the initial state must be a closed theory [14].

Brahms

Brahms is a multi-agent modeling tool to develop models of human and machine behavior. The original purpose of Brahms was to be used in analysis and design of organizations and defining work processes. It is programming language based on rules and is similar to Belief-Desire-Intention (BDI) architecture and other agent-oriented languages, but is based on a theory of work practice and situated cognition. Brahms supports representation of activities which are happen in context of a specific situation in a geographical model of the world [15]. This allows the development of an intelligent agent which is possible to act and react to a specific situation that occurs during its execution [16]. The objective of Brahms is to represent of people's teamwork, off-task behaviors, multi-tasking, interrupted and resumed activities, informal interactions and knowledge, which are all work practice of an organization [15].

GOAL

Goal-Oriented Agent Language (GOAL) is a high-level agent programming language for development of rational agents capable of decision making to satisfy their beliefs and goals. GOAL agents are called rational because they fulfill various fundamental rationality constraints and because they choose to perform activities to further their objectives based upon a thinking plan determined from practical reasoning [17]. The most important characteristic of the GOAL is its concept of declarative objectives and the way agents determine their decision from such objectives. Furthermore, GOAL gives intends to agents to focus on particular goals and to convey at the learning level. The fundamental features of GOAL are: declarative beliefs, declarative goals, blind commitment strategy, rule-based action selection, policy-based intention modules, and communication at the knowledge level [17].

JIAC Agent Platform

JIAC (Java-based Intelligent Agent Component ware) is a Java-based framework suitable for developing distributed large-scale applications which supports design, implementation and deployment of agent programs. The whole development process, from initial phases of requirement analysis to deployment of the application is supported by JIAC; with reusability and run-time modification of applications and services support. Distribution, scalability, adaptability, and autonomy are the most important part of JIAC [18]. Service Oriented Architecture and agent technology are combined together by JIAC [19] and applications can be developed using a prepared library which consists components, services and agents that can be integrated to develop an application capable of performing standard tasks. Other functionalities which are specified by the developer can also be interactively integrated [18]. In this framework, the emphasis is on industrial business requirements such as software standards, security, management, and scalability and it has been developed within industry- and government-funded projects [19].

Agent Factory

The Agent Factory Framework is an open source collection of tools, platforms, and languages that support the development and deployment of multi-agent systems. It is divided into two sections: (1) support for deployment of agents on servers and desktop systems which is realized through Agent Factory Standard Edition (AFSE); and (2) support for deployment agents on devices such as sensors and mobile phones which is realized using Agent Factory Micro Edition (AFME) [20]. Agent Factory will not force the developers to use pre-existing agent architectures and they are free to either use them or develop custom solutions which are suitable for their goal. Agent Factory uses Agent Factory Agent Programming Language (AFAPL), which is a programming language for developing agents with support of agents with intelligent architecture which are able to think about how to play their role. [21].

3. DISCUSSION

All the methodologies reviewed in this paper support basic agent-oriented concepts (Reactivity, Proactivity, and Autonomy). The development process is clarified in all of them and they cover all phases of development from requirement gathering and analysis to detailed design. However Tropos have also an attention to the implementation phase as well, but agent-oriented methodologies still need to cover quality assurance and system management in order to be adopted by industry.

Moreover, comparison of languages/frameworks reviewed in the paper shows that all of them supports concepts such as reactivity, environment awareness, social interaction, mental attitudes, and etc. While one-to-one messaging between agents is supported by MetateM, and Brahms, GOAL, JIAC, and Agent Factory provide facilities to broadcast messages to multiple agents. Since the languages are mostly near to concepts used every day by humans, they are easy to learn and understand. Only in JIAC the programmer needs to be familiar with C, because the syntax style is similar to C programming language. All the languages/framework come with suitable documents which makes the deployment process easier, however languages based on Java (GOAL, JIAC, and Agent Factory) provides more portability. Mobility framework is not built in any of the languages except for JIAC and Agent Factory; however they also have their own limitations in mobility. Table 1 summarizes this comparison based on six criteria as below:

	Support Agent Concepts	**Agent Interaction**	**Support to Design Mobile Agents**	**Easy to Use**	**Deployment and Portability**	**Tool Integration**
MetateM	✓	✓	×	✓	✓	✓
IndiGolog	✓	×	×	✓	✓	✓
Brahms	✓.	✓	×	✓	✓	✓.
GOAL	✓	✓	×	✓	✓	✓
JIAC	✓	✓	✓	×	✓	✓
Agent Factory	✓	✓	✓	✓	✓	✓

TABLE 1: Comparison of AOP Languages/Frameworks.

4. CONCLUSION

In developing an agent which is intended to be introduced to industry, the engineering approach in development is considered as a basic required factor. Therefore methodologies will take a significant role especially when number of agents in a system increases. Thus during development of an agent-based system, management, testing and reusability techniques should be applied. This paper reviewed some methodologies and development languages/frameworks for developing agent-based systems. The methodologies and languages in developing agents can be chosen based on the system requirements, type of agents, and the environment in which agents are being used.

5. REFERENCES

[1] Y. Shoham, "Agent-oriented programming," *Artificial Intelligence,* vol. 60, pp. 51-92, March 1993.

[2] J. Tweedale and L. Jain, "Agent Oriented Programming," in *Embedded Automation in Human-Agent Environment*. vol. 10, ed: Springer Berlin Heidelberg, 2012, pp. 105-124.

[3] M. Winikoff and L. Padgham, "The Prometheus Methodology," in *Methodologies and Software Engineering for Agent Systems*. vol. 11, F. Bergenti, M.-P. Gleizes, and F. Zambonelli, Eds., ed: Springer US, 2004, pp. 217-234.

[4] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos, "Tropos: An Agent-Oriented Software Development Methodology," *Autonomous Agents and Multi-Agent Systems,* vol. 8, pp. 203-236, 2004/05/01 2004.

[5] A. Fuggetta, "Software process: a roadmap," presented at the Proceedings of the Conference on The Future of Software Engineering, Limerick, Ireland, 2000.

[6] E. Yu and L. M. Cysneiros, "Agent-oriented methodologies-Towards a challenge exemplar," in *Proc of the 4 Intl. Bi-Conference Workshop on AOIS, Toronto*, 2002.

[7] L. Padgham and M. Winikoff, "Prometheus: a methodology for developing intelligent agents," presented at the Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1, Bologna, Italy, 2002.

[8] S. A. Deloach, M. F. Wood, and C. H. Sparkman, "Multi-agent Systems Engineering," *International Journal of Software Engineering and Knowledge Engineering,* vol. 11, pp. 231-258, 2001.

[9] K. Dam and M. Winikoff, "Comparing Agent-Oriented Methodologies," in *Agent-Oriented Information Systems*. vol. 3030, P. Giorgini, B. Henderson-Sellers, and M. Winikoff, Eds., ed: Springer Berlin Heidelberg, 2004, pp. 78-93.

[10] N. Alechina, N. Bulling, M. Dastani, and B. Logan. (2012, May 20). *Logics and Multi-Agent Programming Languages*. Available: http://www.agents.cs.nott.ac.uk/events/lmapl-12.

[11] M. Fisher and A. Hepple, "Executing Logical Agent Specifications," in *Multi-Agent Programming:*, A. El Fallah Seghrouchni, J. Dix, M. Dastani, and R. H. Bordini, Eds., ed: Springer US, 2009, pp. 1-27.

[12] H. Barringer, M. Fisher, D. Gabbay, G. Gough, and R. Owens, "MetateM: An introduction," *Formal Aspects of Computing,* vol. 7, pp. 533-549, 1995/09/01 1995.

[13] G. Giacomo, Y. Lespérance, H. Levesque, and S. Sardina, "IndiGolog: A High-Level Programming Language for Embedded Reasoning Agents," in *Multi-Agent Programming:*, A. El Fallah Seghrouchni, J. Dix, M. Dastani, and R. H. Bordini, Eds., ed: Springer US, 2009, pp. 31-72.

[14] G. D. Giacomo, Y. Lesperance, H. Levesque, and R. Reiter. (2001, May 25). *IndiGolog Overview* Available: http://www.cs.toronto.edu/~alexei/ig-oaa/indigolog.htm.

[15] M. Sierhuis, W. Clancey, and R. J. Hoof, "Brahms An Agent-Oriented Language for Work Practice Simulation and Multi-Agent Systems Development," in *Multi-Agent Programming:*, A. El Fallah Seghrouchni, J. Dix, M. Dastani, and R. H. Bordini, Eds., ed: Springer US, 2009, pp. 73-117.

[16] W. J. Clancey, M. Sierhuis, R. v. Hoof, and M. Scott. (2012, May 25). *What is Brahms?* Available: http://www.agentisolutions.com/brahms.htm.

[17] K. Hindriks, "ProgrammingRationalAgents in GOAL," in *Multi-Agent Programming:*, A. El Fallah Seghrouchni, J. Dix, M. Dastani, and R. H. Bordini, Eds., ed: Springer US, 2009, pp. 119-157.

[18] J. Community. (2014, May 25). *JIAC V*. Available: http://www.jiac.de/agent-frameworks/jiac-v/.

[19] B. Hirsch, T. Konnerth, and A. Heßler, "Merging Agents and Services — the JIAC Agent Platform," in *Multi-Agent Programming:*, A. El Fallah Seghrouchni, J. Dix, M. Dastani, and R. H. Bordini, Eds., ed: Springer US, 2009, pp. 159-185.

[20] R. Collier. (2014, May 29). *Agent Factory*. Available: http://www.agentfactory.com/index.php/Main_Page.

[21] C. Muldoon, G. P. O’Hare, R. Collier, and M. O’Grady, "Towards Pervasive Intelligence: Reflections on the Evolution of the Agent Factory Framework," in *Multi-Agent Programming:*, A. El Fallah Seghrouchni, J. Dix, M. Dastani, and R. H. Bordini, Eds., ed: Springer US, 2009, pp. 187-212.