International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 5, Issue 2, ISSN No. 2455-2143, Pages 353-355
Published Online June 2020 in IJEAST (http://www.ijeast.com)

ROLE OF MULTI-AGENTS IN SMART CITY – A STUDY

P.FELCY JUDITH
Associate Professor
T John College, Bangalore

Abstract - Multi Agent Technology has advanced significantly since its initiation and the research area has been gaining its importance recently due to the need for managing intelligence in field of Internet of Things (IoT). This paper introduces the usage of multi-agents framework in order to help agents to interact and co-ordinate to arrive at necessary decision and later to respond based on the arrived decisions.

Keywords **-**Agent; Multi-Agents; Modeling Agents; Agent Oriented Software Engineering;

I. INTRODUCTION

Multi-Agents are derived from distributed artificial intelligence, provide more modular solutions. Multi agent system provides following positive features

Modularity – even though agent design plays primary role in managing modularity, their distributed nature and features like adding run time behaviors and their reasoning to understand the environment and adapt itself to the environment translates itself in to more modular being. The modularity or separation of concern is about the loose coupling of their behavior and about the intelligence they show in performing the tasks.

Reusability – as discussed earlier they have features like adding behavior in runtime.

They use delegates and events to achieve these features, using this feature a tested behavior could be either configured or could be adopted in runtime directly. Multiple agents could easily share same or similar behaviors. This way the reusability is very high with the usage of multi-agents technology.

Runtime integration – basically objects of available in software would be tightly integrated using inheritance or loosely integration using interface implementation. Agents provide additional feature of integrating components in runtime. This could be done using configuration files or by applying rules during runtime. This way the agents could be loaded with various behaviors in the runtime.

Team work – agents are more than objects and components. They have behaviors which could be plugged in during runtime. Also they integrate or cooperate with each other depending on the rules provided to them. These rules could also be configured in runtime. Rules generally do not specify single action to be performed alternately it provides list of actions to the agents. Depending on the situation of the agent environment the agent would perform one of the actions configured through rules.

Intelligence – as specified earlier the agents would be configured with possible rules and depending on the environment the agent executes one of the rules specified or the combination of rules if sufficient information is provided to the agent. This way the agent works intelligently in comparison to the objects or components. Also the agent rules could be provided using various mathematical models as well. This is outside the scope of this research.

II. DOMAIN SPECIFIC LANGUAGES

A domain-specific language (DSL) is a high-level software implementation language that supports concepts and abstractions that are related to a particular (application) domain. A DSL is a language, that is, a collection of sentences in a textual or visual notation with a formally defined syntax and semantics. The structure of the sentences of the language should be defined by means of a grammar or meta-model, and the semantics should be defined by means of an abstract mathematical semantics, or by means of a translation to another language with a well understood semantics.

DSL are designed to solve domain problems and to increase productivity in software development process and to move the requirements closer to the

International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 5, Issue 2, ISSN No. 2455-2143, Pages 353-355
Published Online June 2020 in IJEAST (http://www.ijeast.com)

product. Domain specific languages are equivalent to assembly lines in automobile industry [4]. Domain-specific languages are also called as application-oriented [8] or special purpose [9] or specialized [5] or task specific [7] or application languages [10]. These little languages do not include many features available in the general purpose languages (GPLs) [6].

DSL are also classified into Textual and Graphical. Textual DSL could be constructed using a parser-generator or writing configuration code in the host language and by using XML. All these methods help define a language which would be useful to the later problems. The Graphical DSL has to define notation, domain model, generation, serialization and tool integration. Notation is where we define building blocks with various kinds of shapes and connectors. The shapes and connectors use decorators to display shapes and texts graphically.

Graphical DSL will help capture the framework to define processes and to integrate the involvement of agents. Textual DSL could be used for various needs including communication between agents.

III. GOAL GRAPHICAL MODEL FOR AGENT FRAMEWORK

Tropos is a software development methodology that uses concepts of actor, goal, and dependency to the model. It involves in early requirement, late requirement, architectural design and detailed design, implementation, unit and integration testing phases. Tropos (Bresciani, Giorgini, Hiunchiglia, Mylopoulos, & Perini, 2004) [11] was originally developed at the Tropos adopts Yu's i* framework (Castro, Kolp, &Mylopoulos, 2002) [12] as the base theory of requirement analysis. The purpose of this "system" actor is to provide system operational services to actors depending on services from the last analysis phase. In the detailed design phase, more explicit scenarios of Agents are depicted. Finally, in the implementation phase, Notations used in Tropos can be seen as mental ones, such as goals and tasks (plans) (Perini, Bresciani, Giunchiglia, Giorgini, &Mylopoulos, 2001) [3]. The notations used throughout the analysis and design phase help preserve the semantic mapping. The Tropos provides necessary tool support in every phase of the development.

The Tropos domain model consists of actor diagrams and goal diagrams. Actor diagrams represent resources used by actors. Goal diagrams include dependencies between actors of the system and reflect system architecture. The early requirements analysis, involves the modeling the intentions of the stakeholder. Stakeholders are modeled as actors. The dependencies between actors are modeled as goal, soft goal, task or resource.

A soft goal is a goal without proper clarity, used to specify non-functional requirements. The actor diagram is expanded considering the individual perspective of the independent actor. Goals are decomposed into sub-goals. In the late requirements stage the stakeholder goals are modeled as a new actor. Every individual goal at higher level is decomposed into detailed lower level goals to completely capture the requirement.

Architectural and detailed design involves in identifying and adopting proper architectural styles and design patterns to implement the captured goals of the actors. Architectural design consists of activities to decompose and refine the actor diagram, identifying actors and replacing actors with agents. Detailed design involves itself towards BDI agent architecture. As the result it produces capability diagrams, plan diagrams, agent interaction diagrams to depict protocol interaction and UML class diagram with the modeling information.

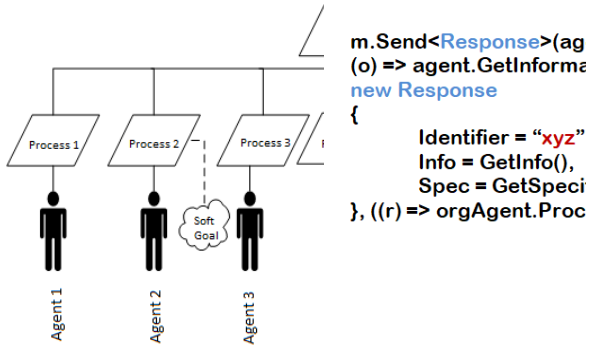

Figure 1: Goal model for Agent Framework, Communication DSL statement

Figure 1 shows the Goal model for Agent Framework and how the agents could be configured using graphical DSL. These agents will communicate between each other using textual DSL developed will look like as given earlier. Individually each of the agents will provide necessary information and

collectively the agent(s) organize this information to take necessary decision and later use other agents to post its response based on the decision made.

IV. CONCLUSION

In this paper we discussed how Multi-Agents could help resolve the complexity involved in IoT by configuring various devices, sensors and actuators into multiple agents and to integrate the agents using graphical DSL and communicate using textual DSL in order to arrive at decisions based on information provided by individual agents and respond to decision using other agents.

V. REFERENCES

[1] Svítek, M.(2014). Telematic approach into program of smart cities. In: EATIS 2014, Valparaiso, Chile.Google Scholar

[2] Přibyl, O., Svítek M(2015).System-oriented approach to smart cities. In: Proceedings of the First IEEE International Smart Cities Conference. IEEE Systems, Man, and Cybernetics Society, New York ,Google Scholar

[3] Perini, A., Bresciani, P., Giunchiglia, F., Giorgini, P., &Mylopoulos, J. (2001), 28 May - 1 June 2001). *Aknowledge level software engineering methodology for agent oriented programming.* Paper presented atthe Fifth International Conference on Autonomous Agents, Montreal, Canada.

[4] P. C. Smolik, (2006). Mambo Metamodeling Environment, A dissertation submitted in partial fulfillment of the requirements for the degree of doctor of philosophy, Brno University of Technology, Czech Republic.

[5] Elizabeth A. Kendall (1999). Role modeling for agent system analysis, design,and implementation. In ASA/MA, pages 204–218. IEEE Computer Society.

[6] BENTLEY, J. L. (1986). Programming pearls: Little languages. Comm.ACM29, 8 (August), 711–721.

[7] NARDI, B. A. (1993). A Small Matter of Programming: Perspectives on End User Computing. MITPress.

[8] SAMMET, J. E.(1969). Programming Languages: History and Fundamentals. Prentice-Hall.

[9] WEXELBLAT, R. L., Ed. (1981). History of ProgrammingLanguages.Academic Press.

[10] MARTIN, J. (1985). Fourth-Generation Languages.Vol. I: Principles, Vol II: Representative 4GLs.Prentice-Hall.

[11] Bresciani, P., Giorgini, P., Hiunchiglia, F., Mylopoulos, J., & Perini, A. (2004). Tropos: An agent-oriented software development methodology, technical report #dit-02-0015. *AAMAS Journal, 8*(3), 203-236.

[12] Castro, J., Kolp, M., &Mylopoulos, J. (2002). Towards requirements-driven information systems engineering: The tropos project, information systems. Elsevier, Amsterdam, The Netherlands.