pISSN: 2523-5729 eISSN: 2523-5739

Open Access
Full Length Article

ORIGINAL ARTICLE

Formal Validation of an Agent-based Location Guidance System (LGS)

Nadeem Akhtar[1], Dost Muhammad Khan[2], Malik M. Saad Missen[3]

[1, 2, 3] Department of Computer Science and IT, The Islamia University of Bahawalpur, Pakistan

ABSTRACT

A multi-agent system has a number of design abstractions as compared to traditional distributed systems. As these systems are distributed, complex and often have a dynamic environment, the formal specification of these systems plays a fundamental role in system correctness. A user location guidance system, based on software agents is designed and verified using formal methods and techniques. In this system, multiple interacting agents work actively in a collaborative way to solve a complex problem. The agent-based location guidance system informs the user about its current location and guides him to find the desired location. We have analyzed the development process after classifying it in the major phases of architecture specification and formal verification. The emphasis is on mathematical based model checking to specify as well as verify agent's behavior.

Keywords: Multi-Agent System; Location Guidance; Formal verifications; Architecture specifications; Correctness; Architecture Description Languages (ADLs); Finite State Process (FSP); Labelled Transition System (LTS); Labelled Transition System Analyzer (LTSA).

Author`s Contribution
[1] Data analysis, Data collection, Conception, interpretation and manuscript writing
[2] Conception, synthesis, and manuscript writing
[3] Conception, Interpretation, and discussion

Address of Correspondence
Nadeem Akhtar
Email: nadeem.akhtar@iub.edu.pk

Article info.
Received: July 26, 2017
Accepted: November 25, 2017

Cite this article: *N Akhtar, D.M. Khan DM, Saad Missen MM. Formal Validation of an Agent-based Location Guidance System (LGS). J. inf. commun. technol. robot. appl.2017; 8(2):53-58.*

Funding Source: *Nil*
Conflict of Interest: *Nil*

INTRODUCTION

Software engineering has developed an increasingly powerful array of methods, techniques, and tools to analyze and design complex software systems; and one of these systems is the idea of intelligent agents and multi-agent system. The agent is a computer system that is capable of acting independently from its users or owners and objective of these agents is to attain the desired goals even without the supervision of their users. An agent is autonomous; it can work independently in its environment; it can make its decisions without the supervision of user [21].

When an agent-based approach is adopted it involves multiple agents. The distributed nature of multi-agent approach solves the problems of multiple logics and multiple interests. In addition, agents interact with each other to achieve their goals and manage dependencies. These interactions shared environment allows agents to interact with the ability to negotiate cooperation, coordination and action plan. "A multi-agent system is a collection of several interacting agents in which each agent has incomplete information or capabilities for solving the problem"[8].

Labeled Transition System (LTS) based on Finite State Process (FSP) provides the ability to formalize the specifications of agent components and architecture. A process consists of one or more actions and is represented by a finite number of states. Formal verification ensures that the developed system is correct and system meets its purpose of development [1].

The Architecture is specified with the help of Architecture Description Languages (ADLs) which provides formal architecture specifications of the system in terms of agents. These formal architecture specifications describe how agents will behave and interact with each other in a system [17].

The Location Guidance System is a platform-independent agent-based application that can run on smartphone, tablet, any hand-held device and laptop. It has a system for Global Positioning System (GPS). It informs a user about their current location and guides them to find the desired locations (i.e. faculties, departments, administrative offices, hotels, banks, mosques, cafeterias, libraries, study-parks etc.) inside the Baghdad-ul-Jadeed campus of The Islamia University of Bahawalpur. This campus is built on 1700 acres of land. Through interacting with the system, user can find out its exact current location along its x-axis or y-axis. The proposed system which consists of autonomous agents may have multiple agents.

Agent-based systems are specialized systems consisting of autonomous agents. These systems are distributed; have greater complexity; have high levels of abstraction, and are concurrent. Due to the above properties the formal analysis, formal modeling, and formal verification of agent-based systems is of critical importance. The problem statement is *formal specification and analysis of an agent-based location guidance system; verification of safety and liveness properties of correctness by a model based on labeled transition system.* The major objective is: to propose a working architecture of an agent-based location guidance system; to propose formal specifications of this system and formal verification of the correctness properties of this system.

The rest of the paper is organized as follow: In section 2 Literature is reviewed, in section 3 Materials and Methods - User Location Guidance System: A Case Study is presented, in section 4 Results are discussed and finally the Conclusion is drawn in section 5.

LITERATURE REVIEW

2.1 Multi-agent system

"An agent is a computer system that is capable of acting independently (i.e. autonomously) for its owner and objective of an agent is to achieve desired goals of the designed system as instructed by its user".[21] [9] "An agent is considered as a computer system situated in some environment, capable of autonomous actions in this environment in order to meet its design objectives" [15] [9]. "A multi-agent system is a collection of several interacting agents in which each agent has incomplete information or capabilities for solving the problem" [8] [10] [11]. Multi-agent systems provide constructs to resolve a problem by decomposing the system in to set of autonomous entities; these autonomous entities incorporated in a multi-agent environment interact (i.e. cooperate, coordinate, collaborate) with their environment and among themselves, and as a result, meet the system quality requirements and functionalities [20] [11] [10] [13] [12]. According to Shardlow, "there is something more to an agent than its capacity for beliefs and desires, but that thing has no unified account within cognitive science. Agents do things, they act: that is why they are called agents" [19].

A multi-agent system can be used to tackle the complexity of software systems and the most recent additions are the ideas of an intelligent agent and multi-agent system. In recent years, the agent technology has proven to be a reliable solution in areas such as tourism, transport, information management, and transport management. A software agent is a piece of code that can interact with its environment to fulfill the requirements independently. Agents perform actions and can take initiative to achieve their objectives.

Unlike software objects, agents are active entities that take an active role in originating actions that affect their environment. Actions originating from agents are described by the terms *autonomy* and *rationality*. Autonomy states that an agent can work without human (or other) supervision, intervention, or guidance. Rationality is often used in the sense of an agent maximizing its performance with respect to some “valuation function”.

2.2 Formal methods

Formal methods are based on mathematics and precisely describe system properties. They provide a framework to develop and verify a system in a systematic way. Formal specification and modeling of a system are fundamental in ensuring correctness, rigor, preciseness, and reliability of the system. More user-friendly methods of specifying limitations and system features are needed so that developers can participate in validation of formal program requirements. Formal methods, techniques, languages, and tools design and develop a system with a well-defined syntax and semantics; provide exhaustive guidelines for using the mathematical notation; provide techniques for analyzing specifications expressed in mathematically based notations.

ADLs are formal languages for a precise and accurate description of multi-agent software architecture in terms of components and connectors; they specify the behavior aspects of components and connectors; describe their composition, and describe their coordination [17] [16]. In future, we have the plan to specify formal architecture of the system and translate it into a working prototype.

2.3 Model Checking

Formal verification ensures the correctness of the system and validation attests whether the designed system meets its intended purposes [1]. Model checking [6] [5] [4] [3] [18] is one of the most widely used techniques for formal verification and validation, it gives a step by step process for a comprehensive survey of the mathematical model. It involves testing all states and transitions in the model. It produces an abstract representation of a system to determine the effect of the system properties. Model checking has been widely used in industrial projects i.e. hardware circuit's verification, verification of communication protocol specifications, verification of aeronautical, spacecraft and satellite software specifications. Its overall aim is to increase the quality of verification and validation by specifying and checking properties that cover all of the application requirements. The primary advantage of model checking is that it is often fully automated.

We have used model-checking based approach for the formal verification. Labeled Transition System (LTS) is a state machine based formalism and is used for behavioral modeling. The system is specified in Finite State Process (FSP) language which generates a labeled transition system. Labeled Transition System Analyzer (LTSA) verifies that the properties and characteristics of the developed system are according to the behavior requirements.

2.4 Labeled Transition Systems (LTS) [14]

Finite State Process (FSP) is the notation used to specify the behavior of concurrent systems. It checks the mechanical specifications of the system while satisfying the requirements and behavior of the system. FSP specifications generate a Labeled Transition System. It describes functional securities of proposed system using the logic that relates to its input and output. It is used for the correct and consistent representation of functional securities. FSP is based on algebra. It is a formal language that can be applied to a system at different levels to implement the formal specification. It provides a discrete system, consisting of parallel processes with synchronization between these processes by action sharing.

2.5 Safety and a Liveness property

Safety property ensures that "something bad that does never happen". In case of user location guidance system, it ensures that user gets desired location service. Safety properties are defined as a deterministic finite state process which is called property automata. Safety property ensures that behavior of transaction is valid or not. "ERROR conditions work like exceptions in programming languages. They show the state that is not required. In complex systems, we specify safety properties by directly stating what is required" [14]. Liveness property ensures that any service execution must reach its desired destination. Liveness property essentially ensures that each client can perform its own operation without knowing about other clients facing any failure or success in performing their actions. It is a functional property. It should always be in the case that service either eventually replies or aborts.

RESEARCH METHODOLOGY

3.1. System Architecture

System architecture describes the components, connectors, the relationship between components,

Table 1: Detailed Working and Abbreviation of Agents used in System	
Agent Name	**Description**
TOURIST AGENT	An instance of a tourist agent is created for each user. They provide services by working with the Database Agent that makes sure the model of the system is being monitored by the Database Agent. When Database Agent made any change in the database, Tourist Agent changes its own database with the correct data.
LOCATION AGENT	Locating user position is the basic function of the Location Agent. Location agent uses Cartesian coordinates to locate and guides the user towards the other locations.
INTERFACE AGENT	Through Interface Agent user can get its initial point to move further. It provides the facility for initial registration. Each user has its own Tourist Id for contacting with Database agent for fetching its location.
DATABASE AGENT	The Database agent manages all the resource information (i.e. hostels, faculties, departments, school, hospital, and mosque). It contains the information of user's location along their X-axis (latitude) and Y-axis (Longitude). All agents (LA, TA, PA, IA, RA) interact with database agent that is connected to a database for information retrieval.
REPORT AGENT	Report Agent creates statistical reports. Reports highlighting a number of searches, number of locations, new locations added into the system etc.

structure as well as behavior of the system. It is "the fundamental organization of a system embodied in its components, their relations to each other, and to the environment, and the principles guiding its design and evolution" [7].

Our system design gives us a formal perspective of the system. It provides awareness about the design of the program, such as program components, visible features of these components and structured relationships. We purpose a formal architecture. This architecture, identify all architectural elements. Architectural elements are identified separately and then associated with the representation of the system as one unit.

In the requirements area, both the services provided to its users and the inner functions of the organizations are explained. The architecture shows the functions of the system. To create a model we start with the analysis of the existing system, re-apply the modeling process with some new functionality to create a system and also integrate the user requirements for the new system. Structure information provides formal requirements of the structure with regards to elements and connections and how they are consisting together at runtime [16].

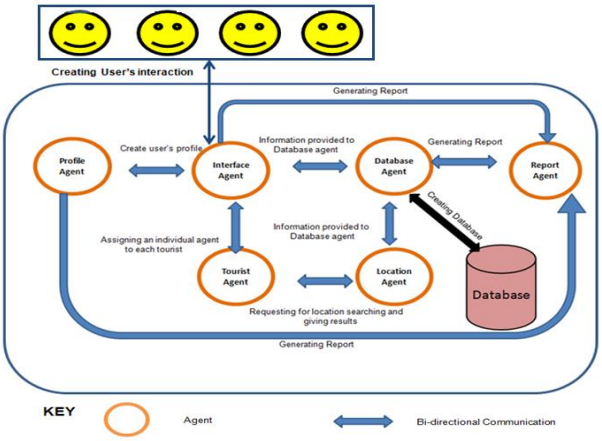

Fig. 1. A Multi-Agent based architecture of User Location Guidance System

The result after tracking will be verified and a statistical report will be generated by report agent. Database agent manages the processing of data related to the exact location and all information of users will be stored in the customer profile and administration. Formal verification will be performed at each level of the proposed system by using LTS. Formal specifications are concise, unambiguous, well-defined and stable. These techniques will help us to gain more knowledge about the system design, to remove ambiguities, to maintain levels of abstraction and also determine our problem [2].The system architecture consists of Tourist Agent, Location Agent, Profile Agent, Interface Agent, Report Agent and Database Agent. Each representative agent receives data and information from database using its database agent. Detailed working of all the agents are listed in table-1.

3.2. A Scalability Agent

The Scalability agent ensures how many agents are working at the same time. The scalability agent has been specified by taking account of the number of running agents at a time.

```
1  // Safety property that ensures Scalability
2  rang N = 1..100;
3  property SCALABILITY = (agent[s:N] -> SCALABILITY
4  | request_agent -> checking_availability -> AGENT_ASSIGNMENT),
5  AGENT_ASSIGNMENT =
6        ( agent[s:N] -> agent_assign -> AGENT_ASSIGNMENT
7  ).
```

3.3. Property NOLOSS_DATA

NOLOSS_DATA is the safety property of information agent. It specifies that there is no loss of data during the steps of agent gathering data from the user, modifying data, and moving data between different agents. All information about user id, user current location, and new location with coordinates is stored.

```
1  // Safety property to ensure that there is no loss of data
2  rang N = 1..100
3  property NOLOSS_DATA = (agent[s:N] -> NOLOSS_DATA
4                        | required_data -> system_checking -> DATA),
5   DATA = ( fetching_data[s:N] -> gethered_data -> DATA ).
```

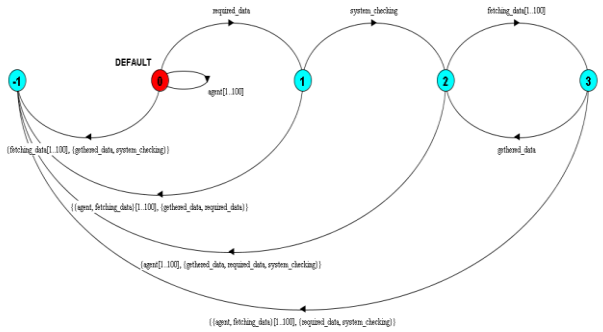

Fig. 2. LTS showing safety property that ensures that there is no loss of data

3.4 Property RANGE

This safety property creates a check on the range of location which is entered by the user in term of service.

```
1  // Range Agent declarations
2  property RANGE = (user_range -> checking_range ->
3                   ( range_valid -> reply -> RANGE
4                   | range_invalid -> reply -> RANGE)
5  ).
```

3.5 Database Access Agent

From the perspective of providing security service to the database; this safety property provides security checks on whether the user is authorized or not. For accessing the proposed system user required to register them. All information of the user stored in database and user can easily access the database if the user is authorized. This FSP checks safety property of USER_ACCESS. When the user wants to access the system then the system checks user authorization. If the user is authorized or registered then access will be granted and if the user is unauthorized then an access denied.

```
1  // Safety property that ensures that the user is an authorized user
2  property DB_ACCESS = (user_access -> access_checking ->
3                   ( authorized -> db_access -> granted -> DB_ACCESS
4                   | not_authorized -> reply -> denied -> DB_ACCESS)
5  ).
```

3.6 Database Agent

Database Agent forms a communication bridge between all agents and the database. The database stores the detailed maps of the University campus. These maps are continuously updated. The user position is detected and services are provided to the user.

```
1  // Database declarations
2  DATABASE_AGENT = ( service.request -> location.track ->
3       ( location_Confirm -> service.reply -> DATABASE_AGENT
4       | location_notConfirm -> service.abort -> DATABASE_AGENT)
5  ).
```

3.7 Tourist Agent

Each user is allocated a separate tourist agent. It provides services by working with Database Agent. The environment maps and details are supervised by the Database Agent and it keeps the database up to date with detailed maps.

```
1  // Tourist Agent declarations
2  TOURIST = (request_agent -> checking_availability ->
3  (agent_assign -> TOURIST | agent_notAssign -> TOURIST)
4  ).
```

CONCLUSION

The proposed user location guidance system is specified and modeled by using multi-agent system technology. A multi-agent system is inherently complex, distributed, having dynamic environment shared by all agents. A fundamental concern is to ensure safety and liveness properties of correctness. These properties play a fundamental role in system correctness. In this paper, a formal model of the user location guidance system is proposed. The purpose of this formal model is the exhaustive investigation of the multi-agent system state space in order to ensure that the safety and liveness properties are held, and there is no single violation of these properties in any part of the system. The proposed model is rigorous as well as exhaustive. The finite state process based notation is used to generate the labeled

transition system model of the user location guidance system.

REFERENCES

1. Bakar, N. A., & Selamat, A. (2011). Analyzing model checking approach for multi-agent system verification. Paper presented at the Software Engineering (MySEC) 5th Malaysian Conference, Malaysia.
2. Bowen, J., & Hinchey, M. (1995). Ten Commandments of Formal Methods. IEEE Xplore, 28(4), 56-63.
3. Clarke, E. M., & Emerson, E. A. (1981). Design and synthesis of synchronization skeletons for branching time temporal logic. In Logics of Programs, LNCS, 131(Yorktown Heights, NewYork), 52-71.
4. Clarke, E. M., Emerson, E. A., & Sistla, A. P. (1986). Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM transactions Prog. Lang. Syst., 08(02), 244-263.
5. Clarke, E. M., Grumberg, O., & Long, D. E. (1994). Model checking and abstraction. ACM transactions Prog. Lang. Syst., 16(5), 1512-1542.
6. Clarke, E. M., Grumberg, O., & Peled, D. (1999). Model Checking.
7. IEEE. (2000). Recommended Practice for Architectural Description of Software-Intensive Systems. In IEEE Standard No. 1471-2000. Available at http://shop.ieee.org/store/. Product No.: SH94869-TBR.
8. Jennings, N. R., Sycara, K., & Wooldridge, M. (1998). A Roadmap of Agent Research and Development. International Journal of Autonomous Agents and Multi-Agent Systems (IJAAMAS), 1(1), 7-38.
9. Khan, D. M., & Mohamudally, N. (2010). An Agent-Oriented Approach for Implementation of the Range Method of Initial Centroids in K-Means Clustering Data Mining Algorithm. INTERNATIONAL JOURNAL OF INFORMATION PROCESSING AND MANAGEMENT, 1(1), 104-113.
10. Khan, D. M., & Mohamudally, N. (2012). Intelligent versus Malicious Agent: A Comparative Study. INTERNATIONAL JOURNAL OF COMPUTER SCIENCE ISSUES (IJCSI), 9(2), 605-611.
11. Khan, D. M., Mohamudally, N., & Babajee, D. (2012). The Formulation of a Data Mining Theory for the Knowledge Extraction by means of a Multiagent System. JOURNAL OF COMPUTING, 4(8), 29-38.
12. Khan, D. M., Mohamudally, N., & Babajee, D. (2012). Towards the Formulation of a Unified Data Mining Theory, Implemented by means of Multiagent Systems (MASs). In A. P. A. K. (Ed) (Ed.), Advances in Data Mining Knowledge Discovery and Applications (pp. 03-42).
13. Khan, D. M., Mohamudally, N., & Babajee, D. (2013). A Unified Theoretical Framework for Data Mining. SciVerseScienceDirect, Information Technology and Quantitative Management (ITQM2013), Procedia Computer Science by Elsevier 17 104-113.
14. Magee, J., & Kramer, J. (2006). Concurrency: State Models and Java Programs: John Wiley and Sons.
15. Michael Wooldridge, N. R. J. (1995). Intelligent agents. Theory and practice. Knowledge Engineering Review, 115-152.
16. Oquendo, F. (2004). π-ADL: An Architecture Description Language based on the Higher-Order Typed π-Calculus for Specifying Dynamic and Mobile Software Architectures. ACM-Software Engineering Notes, 29(03), 1-14.
17. Oquendo, F. (2008). Dynamic Software Architectures: Formally Modelling Structure and Behaviour with Pi-ADL. Paper presented at the Software Engineering Advances, ICSEA '08, The Third International Conference in 2008.
18. Quielle, J. P., & Sifakis, J. (1982). Specification and verification of concurrent systems in CESAR. Paper presented at the Proceedings of the 5th International Symposium on Programming.
19. Shardlow, N. (1990). Action and agency in cognitive science. (Master thesis), University of Manchester, Oxford Rd., Manchester.
20. Weyns, D., Holvoet, T., & Schelfthout, K. (May 2006). Multi-agent Systems as Software Architecture another Perspective on Software Engineering with Multi-agent Systems. Paper presented at the International Conference on Autonomous Agents and Multiagent Systems (AAMAS).
21. Wooldridge, M. (2009). An introduction to multiagent systems: John Wiley & Sons.