Technical Journal, University of Engineering and Technology (UET) Taxila, Pakistan Vol. 25 No. 4-2020
ISSN:1813-1786 (Print) 2313-7770 (Online)

Approach for the Formal Specification, Modelling, and Verification of a Safety-Critical Multi-agent System; An Employees Management Multi-agent System (EM-MAS)

N. Akhtar[1], M. Pasha[2], A. Rehman[3]

[1] *Department of Computer Science and IT, Faculty of Computing, The Islamia University of Bahawalpur, 63100 Pakistan,*
[2] *Department of Information Technology, Bahauddin Zakariya University Multan, 60800 Pakistan,*
[3] *Department of Computer Science and IT, Virtual University of Pakistan, Lahore, 54000 Pakistan,*

[1] nadeem.akhtar@iub.edu.pk

Abstract- In today's advanced software engineering, intelligent agents are getting more popular. Agent-based technology offers a new dimension for designing and implementing intelligent software systems. Intelligent agents are autonomous and they can make decisions with very little or sometimes no supervision. A multi-agent system comprises multiple intelligent agents that exhibit intelligent and autonomous behavior. In addition, intelligent agents work with minimum or no direct human interaction and have control of their actions as well as their internal states. The goal of employees management multi-agent system is to provide principles and mechanisms for constructing a fault-tolerant distributed system. The multiple autonomous software agents can be used for managing the communication, cooperation, and coordination between these autonomous agents and their continuously changing environment. Autonomous multi-agents are used in an employee management system, for a distributed, flexible, adaptable and fault tolerant system. The distributed nature of this system, make it fault tolerant and error-free as it keeps on working if some agents go fail. The Employees Management Multi-Agent System (EM-MAS) is designed, modeled and verified using a formal approach based on mathematical constructs of Regular expression, First-order predicate calculus, functions, relations, and set theory. Gaia multi-agent methodology is being used for specification, analysis, preliminary design and detailed design of the system. It is centered on the idea of a system as a computational organization consisting of interacting agents with each agent having one or more roles. The system is formally modeled and verified by constructing the system in the form of Coloured Petri Nets (CP-Nets). Formal verification in the form of Model checking ensures the correctness properties of the proposed model.

Keywords- Multi-agent system; autonomous agents, Gaia multi-agent methodology; Formal modeling; Formal verification; Model checking, Coloured Petri Nets (CP-Nets); Employees Management Multi-Agent System (EM-MAS).

I. INTRODUCTION

Multi-agent system based software applications are designed and implemented by autonomous agents that can achieve their goals and objectives by working together i.e. agents can cooperate, coordinate, and communicate in order to attain a common objective. Multi-agent environment is an essential part of the multi-agent system as it provides the agent a surrounding environment as well as conditions to exist these interacting agents to exist. The Environment is the medium of coordination between intelligent agents. Several agent-based systems like air traffic control system and Information management systems have revolutionized.

An intelligent agent is autonomous, has the problem-solving capability and is reactive to social behavior in its environment. It can take action on the user's behalf and its tasks include finding and filtering of information, transfer of services, automation of complex tasks and collaboration with other agents for solving complex problems. Agents perform such tasks continuously and autonomously in multi-agent environments [1][2].

The research objective in this work is the formal specification, modelling and verification of an employee's management multi-agent system. As this system is a distributed system based on autonomous agents having concurrent executions, therefore to ensure the correctness properties of this system is challenging. This system manages all the information of an employee (i.e. salary calculation, bonuses calculation, holidays, sick-leaves, performance attributes, domain of expertise, domain of operations,

skill-level, tasks-assigned, tasks-completed, contact information), and an employee is the back-bone for the working of any enterprise therefore the correct working of this system is of critical importance. Thus it is a safety-critical system and without the smooth and correct working of this system the enterprise cannot function at all. The research questions are:

RQ1: What are the challenges in ensuring the correctness properties of safety and liveness of a distributed system based on multiple agents.

RQ2: How it is possible to formally specify and model a multi-agent system like EM-MAS.

RQ3: How it is possible to formally verify the correctness properties of a multi-agent system like EM-MAS.

RQ4: How it is possible to propose an approach based on formal verification i.e. model checking to formally verify the correctness properties of a multi-agent system like EM-MAS.

An employee management system is proposed based on multi-agents. This system is formally analyzed, preliminary and detailed designed and modeled using Gaia multi-agent methodology. The agent roles are specified and developed to show the liveness and safety properties of agents and protocol model is defined for agent interactions. Afterward, these formal design specifications are transformed into formal verification specification and as a result, a model is constructed using the Coloured-Petri Nets (CP-Nets). The CP-Nets model provides rigorous formal modeling and verification of liveness and safety properties and subsequently ensures the correctness of the system.

II. Background Studies

a. Safety and Liveness Properties:

Correctness

Correctness properties of safety and liveness provide exhaustive system verification. Safety and liveness properties both are equally important for the formal verification and validation, they complement each other (i.e. liveness alone or safety alone is not sufficient to ensure exhaustive verification. Both the safety and liveness are functional properties. The liveness property is directly related to the core behavior of the system. The functions and roles of the system are specified as liveness. On the other hand, safety property is used to specify the constraints under which the system has to operate. The quality attributes of the system can also be specified as safety properties. The safety and liveness properties of the system are ensured by formally specifying and verifying these properties in Role model of the Gaia multi-agent methodology and CP-Nets.

Safety property is a deterministic process that models the constraints and limitations of the system and ensures that in a system an acceptable state of affair is maintained. Safety property is ensured in model checking [3]. During the model checking the state space is generated, and state-space analysis is performed, all trace of actions in a state space is analyzed. During safety analysis it is ensured that any trace of action in the state space is a part of state space and do not lead to the ERROR state. An ERROR state in a state space is like an exception. In safety property we specify the conditions which are not required, but in complex systems we specify safety property by stating only the conditions which are required.

According to [4] a safety property is specified in temporal logic, safety formula of the form □p (i.e. temporal operator □ meaning always) . This temporal formula ensures that during the computation the property □p holds.

The liveness property complements the safety property. It ensures that the states of affairs of the system is maintained and the system ensures the behavior properties i.e. functional properties [5].

b. Model Checking

EM-MAS is a safety-critical system, and it is complex and plays important role in saving financial losses of an enterprise. The correctness and reliability of the EM-MAS is important in the design process.

Model checking [6]-[13] is a practical formal method for automatic, systematic and exhaustive verification. Its objective is verification by specifying and automated checking of correctness properties. A mathematical model of the system is created and a comprehensive review of the model is performed. It includes checking all states and transitions in the model i.e. exhaustive analysis of the model. It creates an abstract representation of a system with all the possible states and transitions. The correctness of the model is verified through model checking techniques. The model takes input and systemically checks whether the system properties are ensured by the system. It is widely used in verification of safety-critical systems i.e. verification of software systems of airplanes, spacecraft, trains, metros, satellites, control units of a nuclear reactor, and monitoring systems for earthquakes, tsunamis, floods, etc.

Model checking uses temporal logic [14], linear-time temporal logic (LTL) [15]-[17] and computation-tree temporal logic (CTL) [14] [18] [19], for stating, checking, and verifying behavioral properties.

According to [10] model checking is an automated technique, a formal property is defined, a finite-state model of a system is constructed, then the automated tool model checker automatically and systematically evaluates and verifies that this formal property holds for the model. Model-checking is based on models and in these model-based formal verification techniques the system behavior is described in a mathematical-based unambiguous and precise specifications. As compared to formal specifications, informal and semi-formal system specifications are inconsistent,

Gaia Multi-agent methodology [8][9]	**Analysis phase**	• *Organizational model* • *Environmental model* • *Preliminary roles model* • *Preliminary interaction model* • *Organizational rules*
	Preliminary design phase	• *Roles model* • *Interaction model*
	Detailed design phase	• *Agent model* • *Services model*

ambiguous, and incomplete.
The model-checker implements an algorithm that systematically explores all possible states of the model. Therefore, model-checker gives exhaustive verification and validation as compared to simulation (i.e. restrictive set of scenarios of model) and testing (i.e. restrictive set of test-cases of system implementation)

c. The Gaia Multi-Agent Methodology
Gaia [20] [21] is a methodology for the requirement analysis, organizational rules, design, and detailed design of the system centered on organizational abstractions. The Gaia specification process consists of the following three major phases: analysis, preliminary design, and detailed design. Each of these major phases has sub-phases as listed below:
The computational organization is defined in the analysis phase. The global organizational structure is identified and efficiently decomposed into a loosely coupled sub-organization. Environmental model is specified which is a computational representation of agents environment. Preliminary roles are defined without mapping roles into agents. Basic interactions are identified to accomplish the preliminary roles.
The organizational rules are defined. The system is formed as a result of the organization of agent, and the system globally respects and enforces these organizational roles.
In the preliminary design phase, the agent roles and interactions between agents are defined. The detailed design phase has an agent model which defines the agent classes. The system is constructed by these agent classes, and the agent instances are instantiated from them. There can be a one-to-one or one-to-many correspondence between agents and agent roles. The services model identifies the activities of the agents. These services are critical to fulfilling agent roles.

d. Coloured Petric Nets (CP-Nets)
CP-Nets [22] [23] [24] [25] [26] [27] is a discrete-event formal modeling language with the primary objective to formally model and verify systems in which concurrency and communication are the most important characteristics. CP-Nets has a flow-chart like structure in the form of Petri-nets, as well as functional programming language CP-Net Meta-Language (i.e. ML founded on Standard ML [28] [29]). CP-Net has graphical syntax that is ideal for specifying concurrent systems and analyzing their properties. The most widely used application areas of CP-Nets protocol verification [30], communication networks [31], distributed algorithms [32], embedded systems [33], and manufacturing systems [34].

Formal Definition:
"A ***non-hierarchical Coloured Petri Net*** is denoted as a nine-tuple ***CP-Net = (P, T, A, Σ, V, C, G, E, I)***, where:
1. ***P*** is a finite set of places.
2. ***T*** is a finite set of transitions *T* such that $P \cap T = \varnothing$.
3. $A \subseteq P \times T \cup T \times P$ is a set of directed arcs.
4. Σ is a finite set of non-empty colour sets.
5. ***V*** is a finite set of typed variables such that $Type[v] \in \Sigma$ for all variables $v \in V$.
6. $C: P \rightarrow \Sigma$ is a colour set function that assigns a colour set to each place.
7. $G: T \rightarrow EXPR_V$ is a guard function that assigns a guard to each transition ***t*** such that $Type[G(t)] = Bool$.
8. $E: A \rightarrow EXPR_V$ is an arc expression function that assigns an arc expression to each arc a such that $Type[E(a)] = C(p)MS$, where p is the place connected to the arc a.
9. $I: P \rightarrow EXPR_{\varnothing}$ is an initialization function that assigns an initialization expression to each place p such that $Type[I(p)] = C(p)MS$" [22].

CP-Nets are ideal to specify and model the behavior of real-time concurrent safety-critical systems. A CP-Nets model of a safety-critical system is constructed and the liveness and safety properties of correctness are exhaustively analyzed and verified.

Formal Definition:
"A ***hierarchical Coloured Petri Net*** is a four-tuple

$CPN_H = (S, SM, PS, FS)$ where:

1. S is a finite set of modules. Each module is a **Coloured Petri Net Module** $s = ((P^s, T^s, A^s, \Sigma^s, V^s, C^s, G^s, E^s, I^s), T^S_{sub}, P^S_{port}, Pt^s)$. It is required that $(P^{s1} \cup T^{s1}) \cap (P^{s2} \cup T^{s2}) = \varnothing$ for all $s1, s2 \in S$ such that $s1 \neq s2$.
2. $SM : T_{sub} \rightarrow S$ is a **sub-module function** that assigns a **sub-module** to each substitution transition. It is required that the module hierarchy is acyclic.
3. PS is a **port–socket relation function** that assigns a **port–socket relation** $PS(t) \subseteq P_{sock}(t) \times P^{M(t)}_{port}$ to each substitution transition t. It is required that $ST(p) = PT(p')$, $C(p) = C(p')$, *and* $I(p)\langle\rangle = I(p')\langle\rangle$ *for all* $(p, p') \in PS(t)$ and all $t \in T_{sub}$.
4. $FS \subseteq 2^P$ is a set of non-empty **fusion sets** such that $C(p) = C(p')$ and $I(p)\langle\rangle = I(p')\langle\rangle$ for all $p, p' \in fs$ and all $fs \in FS''$ [22].

The formal description below summarizes the definition of the module hierarchy.

Formal Definition:

"The ***module hierarchy*** for a hierarchical Coloured Petri Net $CPN_H = (S, SM, PS, FS)$ is a directed graph $MH = (N_{MH}, A_{MH})$, where

1. $N_{MH} = S$ is the set of **nodes**.
2. $A_{MH} = \{(s1, t, s2) \in N_{MH} \times T_{sub} \times N_{MH} \mid t \in T^{s1}_{sub} \wedge s2 = SM(t)\}$ is the set of **arcs**.

The roots of MH are called **prime modules**, and the set of all prime modules is denoted by S_{PM}" [22].

CP-Nets efficiently develops and verifies a concurrent system by making an executable model of the system. The method of first developing an executable model of the system and then simulating it results in significant new insights into the requirements, design, and operation of the system. This, in turn, results in a less complex and more perfect design. Moreover, developing an executable model leads to an exhaustive evaluation of the system; detail specifications; systematic investigation of scenarios. This all results into a significant decrease in the number of requirement, design, and implementation errors.

In the early stage of analysis and design, the CP-Net model representing the highest level of abstraction is specified. There can be a number of abstractions in a model, the highest level of abstraction is refined into a more concrete and detailed lower level of abstraction. In a model, it is a good approach to have minimum three to maximum five abstraction level. CP-Nets models can be exhaustively simulated. As a result of these exhaustive simulations of CP-Nets models of different abstraction levels, the correctness properties of the system are ensured; the design of the system is validated; every single step in a simulation is observed and validated.

III. Employees Management System

Introduction

The major objective is the analysis, design, formal modeling, formal verification, and implementation of an Employees Management multi-agent system. These agents of the system are autonomous and can make independent decisions. This system works on a distributed level and each distributed node is consist of an agent. These type of distributed systems are more fault-tolerant as compared to centralized systems. Even if a few agents fail the remaining agents keep on working and the overall system remains in a working state.

This system has a Gaia multi-agent based *preliminary agent role model* as well as a *detailed agent role model*. The roles of each employees management system agents are defined. Interaction model defines the protocols between agents. The detailed role model of Gaia multi-agent methodology is transformed into the CP-Net based formal behavior models for formal modeling and formal verification. The detailed role model, as well as services model, are translated into formal behavior CP-Nets model for verification.

The *Services model* is defined which can be translated into any high-level programming language code. This research work is centered on analysis, preliminary design, and detailed design based on Gaia of the employees' management multi-agent system. The agent role model developed as a result of Gaia is transformed into CP-Net models. These CP-Nets models are formal and they exhaustively specify and verify the safety and liveness properties of correctness.

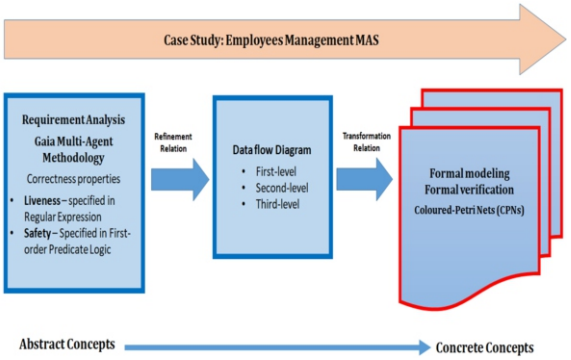


Fig.1. Employees management MAS phases of requirement specifications, correctness properties, data flow and formal verification.

The requirement specifications and verification specifications of the EM-MAS are presented. The EM-MAS consists of agents for managing employees information i.e. attendance, salary, increment, bonus, holidays, leave records etc. The employees' information is stored in the database which is the main environment of the system.

It includes all operations of insertion, deletion, and modification. There are three principal agents

1. *Admin agent:* The Admin agent is responsible for maintaining day-to-day employees, financial, accounting, and administrative services.
2. *Employee agent:* Employee agent manages the employee role in an organization. The employee may be either a permanent employee or on a contract basis or on daily wages. The permanent employee is the regular full-time employees and the contract employees may work for the fixed period which has been mentioned at the time of recruitment i.e. for 5 years and the daily wages employees are the workers that work on daily wage basis and do not get certain facilities which other employees may avail.
3. *Database-Handler agent:* This is responsible for insertion, updates, and deletion of employee records. If the recruitment agent selects an employee for the job then the employee's record containing employee's information is inserted in the database. The department management agent also manages his department in the database; his salary and leave record is also maintained. There should be a guarantee of no duplication of data and records in the system.

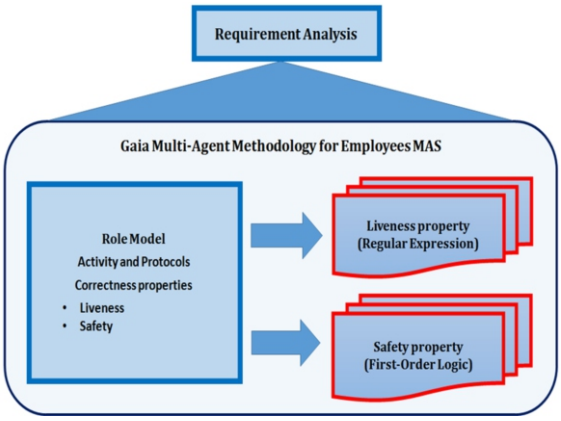


Fig. 2. Requirement specifications and definition of correctness properties of liveness and safety

b. Multi-Agent Environment

Followings are the environment elements for the Employees Management MAS.

1. Database: The overall record of the employees, their salaries and leave records are placed in the database. The database of employees is an organized collection of data consisting of tables, queries, reports, and views.

2. Graphical User Interface: Interface is an important component in the Employees Management MAS through which the records from the database are accessed.

c. Scenarios

In this particular case study, an employee is initially recruited and his record is inserted in the database, his department is also managed, his attendance is marked which may be marked by using a biometric system. The attendance of the employee is managed by the Attendance-Recorder agent, whenever an employee is on leave which may be Casual-Leave, Medical Leave or Earned-Leave or whatever the case may; his leave record is managed by the Leave-management agent. Then at the end of the month, the salary of the employee is calculated by the Salary-calculator agent. All over the record is maintained in the database by the database handler agent.

Gaia method consists of a number of models, we will be using these models for specifying the system and more specifically the correctness properties of liveness and safety of the multi-agent Employee Management System.

d. Organizational Rules

Organizational rules specify the global rules and constraints of the system. These rules and constraints are identified by the analysis phase and are respected by the actual organization of agents. These constraints are satisfied and respected by agent roles and protocols.

e. System Roles

1) Insert Employee Role

It is a role of the Admin agent. Its objective is to add new employees to the employees' management system

Role Schema: **Insert_employee**
Description: Role of an Admin agent in an organization.
Protocols and Activities**:** loginAdmin, receiveRecord , addEmployee, assignId, addPersonalInfo, assignDept, assignDesignation, mentionSalary, assignUserLogin, verifiesUserLogin
Permissions: reads: *employee_id* *employee_login*
Responsibilities: ***Liveness:*** **Insert_employee** = loginAdmin.receiveRecord.(addEmployee.assignId. addPersonalInfo.assignDept. assignDesignation. mentionSalary. assignUserLogin. verifiesUserLogin)
Safety: **∀x ∈ employee_id(x): ¬(employee_id(x) = null)** **∧ ∃x, y ∈ employee_id: ¬(employee_id(x) = employee_id(y))** **∧ ∃x ∈ employee_login ∧ ∃y ∈ employee_login: ¬(employee_login(x) = employee_login(y))**

As shown in the above role, activities (i.e. underlined) are addEmployee, assignId, addPersonalInfo, assignDept, assignDesignation, mentionSalary, assignUserLogin, verifiesUserLogin and there are protocols loginAdmin, receiveRecord.

2) Update Employee Role
It is a role of the Admin agent in order to update the employees' record in the system.

Role Schema: **Update**
Description: This role of the Admin agent which involves updating employees information when required
Protocols and Activities: getsInfo, updateRecord, confirmUpdation
Permissions: reads: *employee_id* changes: *employee_info* *employee_status (external)* /// Checks if the employee is currently working or not.
Responsibilities: ***Liveness:*** **Update** = (getsInfo, updateRecord, confirmUpdation) ***Safety:*** **∃x ∈ employee_id(x): ¬(employee_id(x) = null) ∧ ∃x, y ∈ employee_id: ¬(employee_id(x) = employee_id(y)) ∧ ∃x, y ∈ employee_id: (employee_info(x) = employee_info(y))**

There is only one activity updateRecord and two protocols getsInfo and confirmUpdation. The Admin agent gets employee ID against which employee's record is to be changed; it finds the employees record through employees ID, the employee's information is fetched and changes are made.

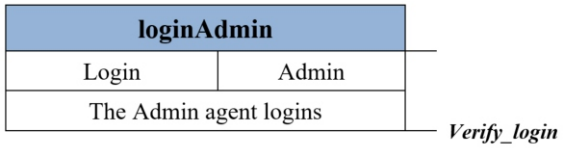
3) Delete Employee Role
It is a role of the Admin agent

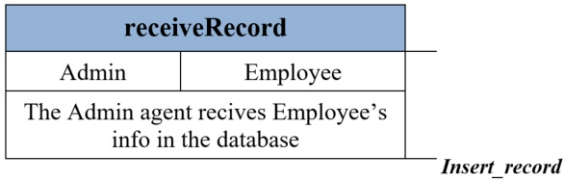
Role Schema: **Delete Employee**
Description: This role of the Admin agent which involves deleting employees information when employee leaves the organization.
Protocols and Activities: getInfo, deleteRecord, confirmDeletion
Permissions: reads: *employee id* changes: *employee info* *employee status = false* /// the employee is not currently working in the organization.
Responsibilities: ***Liveness:*** **Update** = (getInfo, deleteRecord, confirmDeletion) ***Safety:*** **∃ x ∈ employee_id(x): ¬(employee_id(x) = null) ∧ ∃ x ∈ employee_id(x): (employee_id(x) = null) ∧ (employee_info(x) = null)**

The role Delete_Employee has only one activity deleteRecord and two protocols are getInfo and confirmDeletion. Whenever an employee leaves the organization its record must be deleted. The admin agent is responsible for that. The Admin agent gets employee ID against which employee's record is to be deleted and matches it along with the original stored Employees ID if it matches employees info are fetched and deletion is done.

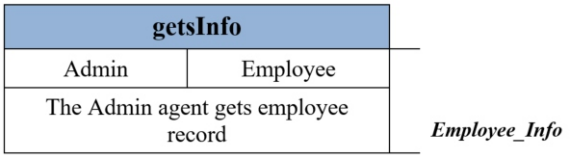
f. Protocols Definitions
Protocols govern the interactions between roles. There exist relationships and dependencies between different roles in a multi-agent organization which is defined as a set of protocols. Here a protocol is a pattern of interaction that is formal and defines a relationship between two roles. Here is the list of the protocol definitions of our system.

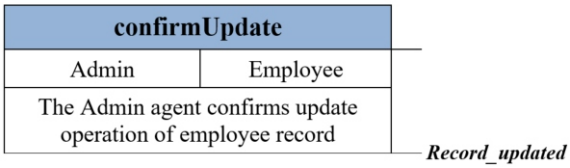
1) Add_Employee



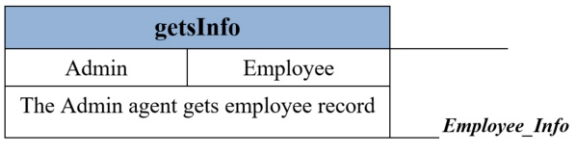


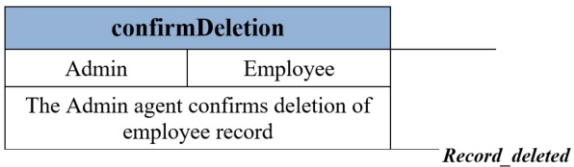
2) Update _Employee





3) Delete_Employee





g. Exhaustive Behavioral modeling based on CP-Nets

Behavioral modeling is important to depict the functionality of the system. It is ideally suitable to model behavior in systems having concurrent executions, or having continuous interactions with user requests, or having interactions with other systems. Further, behavioral modeling helps to assess and modify the system without much cost and effort. It performs functional modeling and starts from the requirement analysis and transform these requirement specifications into implementation. For a distributed system like the proposed EM-MAS, one of the fundamental requirement is to ensure the correctness properties of liveness and safety. Formal modeling exhaustively checks the system behavior and ensures the correctness properties.

CP-Nets based formal specifications are used for the formal specification, modeling, state-space analysis, and model-checking of the employees management multi-agent system. CP-Nets is ideal to model distributed, and concurrent systems like multi-agent system. It is formal, based on temporal logic and set theory. An executable model of the system with all the places and transitions are developed.

CPNTools [36] is an Integrated Development Environment (IDE) to construct CP-Nets model. It allows model-checking, generating state-space of the system, state-space evaluation, and verifying the system by state-space analysis. The state space analysis checks properties if bounded-ness, fairness, liveness, and home [35].

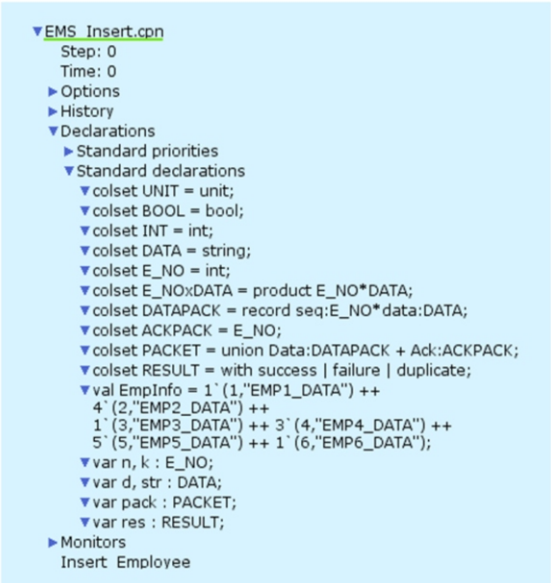


Fig. 3. CP-Nets primitive and composite data types (i.e. CP-nets color sets) for Insert Employee operation.

h. CP-NETS MODEL OF EMPLOYEE MANAGEMENT MAS

Design and workflow structure of the System provides an initial guideline to construct CP-Nets models as it is representing basic functionality of the System. The System is divided into following sub-parts or modules: *Administrator* and *Employee*. CP-Nets models are developed for each of these sub-parts. The analysis and behavior modeling are carried out by using the following steps:

1. Construct CP-Nets models.
2. Run and simulate the model in CP-Nets tools.
3. With the help of state-space tool-kit perform the state-space analysis of the program.
4. After state-space evaluation, if the abnormalities and error arise, errors are analyzed and corrected.

The tool-kit *CPNTools* generates the state space that is analyzed to check the properties of fairness, liveness, home, and bounded-ness. The state space analysis checks and verifies all these properties.

Bounded-ness Properties keeps a track of the number of tokens a place may hold. It also keep a record of the identification of each token.

Home Properties identify the home markings i.e. that can be reached from any state.

Liveness Properties ensures that the transitions are live and can be activated again.

Safety properties assert that something bad will never happen.

CP-Nets Model for Insert Employee data in the system: The CP-Nets model for inserting a new employee in the database of Employees Management multi-agent system is constructed as shown in fig. 4. This model gives a complete overview of inserting a new employee in the employees database

CP-Nets Model for Update_Employee: The states in this model are, inactive, waiting and performing which the Database Manager uses. The message can be in four different states unused, sent, received and acknowledged.

CP-Nets Model for Delete_Employee: The database manager receives a message for deleting information from the Employee Management System, the relevant record is deleted and delete operation is acknowledged.

i. TRANSFORMATION FROM SAFETY AND LIVENESS PROPERTIES OF GAIA ROLE'S MODEL INTO CP-NETS

The liveness properties are expressed by using regular expression and the safety properties are expressed by using first-order predicate logic, in the requirement and design specifications of the EM-MAS. Both these are mathematical constructs, and they precisely and accurately specify the safety and liveness properties. Afterward, these liveness and safety properties are transformed into the CP-Nets. The transformation of these formally expressed properties into a functional language Standard ML of CP-Nets and the graphical

syntax of CP-Nets, is done manually. Afterward, the exhaustive state space evaluation is performed by using the model-checking tool of CPNTools. In the future, we have an objective to design and develop a tool to automate this transformation from Gaia multi-agent methodology role model to CP-Nets.

IV. Results and Discussion

Starting with the requirement elicitation, a multi-agent system is designed at different abstraction levels (i.e. refinement layers). The specified system and the proposed design is modeled formally and CP-Nets models are constructed.

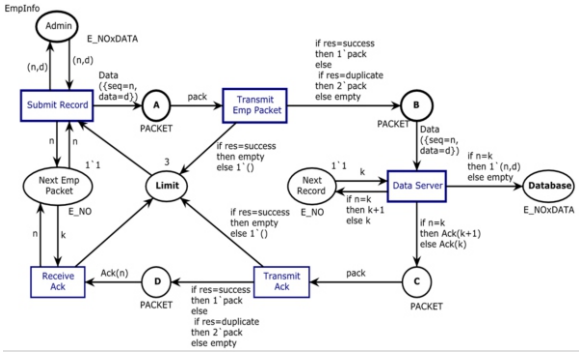


Fig. 4. CP-Nets model for Insert Employee operation of the Employee Management System.

This modeling not only helped to reduce the gap between the requirements elicited textually and formal and graphical notation i.e. CP-Nets. In this study, there are three principal agents that include Admin agent, Employee agent and Database handler agent. The proposed CP-Nets model is consist of three basic functionalities that are Insert_Employee, Update_Employee, and Delete_Employee.

Admin agent has the authority of inserting, updating and deletion of an employee in the database of the system. Fig. 4 described the admin state for inserting employee in the database. It also describes the different states in which an admin agent goes through for insertion operation. For updating record of an employee, admin agent has to go through in different states like Inactive, waiting and performing. The CP-Nets model for updating record having four different states of a message (i.e. sent, unsent, received and acknowledged) is also modeled.

The deletion of an employee's record, as well as the update of employee's record is also modeled and verified.

V. Conclusion and Future Work

Fault-tolerant and error-free software systems are critical for the correctness, reliability, and robustness of the EM-MAS. With progress in science and technology, it is important to design systems that are functionally correct (i.e. error-free) and reliable. The first and foremost objective of EM-MAS is to provide an error-free and fault-tolerant system. The multi-agents in EM-MAS can act autonomously and with little human interaction. These agents work in a distributed environment, which makes this system more reliable. Distributed multi-agents are designed and developed for efficient and fault-tolerant negotiation, coordination, and communication between agents. This agent-based employee management system specifies the formal roles and interaction models. We have designed, modeled, and verified an approach based on mathematics, and more precisely on lambda-calculus as CP-nets Meta-Language (ML) is based on lambda-Calculus. Gaia multi-agent methodology based on *organizational abstractions*, *role*, *protocol*, and *services models* are implemented for formal requirement analysis, preliminary design, and detailed design. The role and protocol models are presented. There is a great advantage of using Gaia for requirements, preliminary design, and detailed design specifications. The use of the regular expression for the definition of liveness properties and the use of first-order predicate logic for the definition of safety properties make these specifications rigorous, accurate, and precise. These Gaia models are then transformed into CP-Nets for the exhaustive formal verification of the correctness properties of liveness and safety. These properties ensure system correctness.

In the future, the goal is to use this proposed approach for the design and development of formal models and formal verification for the dengue and COVID-19 virus information management system.

References

[1] A. Sturm and O. Shehory, “A Framework for Evaluating Agent-Oriented Methodologies,” *Agent-Oriented Information Systems Lecture Notes in Computer Science*, pp. 94–109, 2004.

[2] Y. Shoham, “An overview of agent-oriented programming,” *Software agents*, vol. 4, 1997.

[3] J. Magee, J. Kramer, "State models and java programs", wiley Hoboken, 1999.

[4] Z. Manna, A. Pnueli, "Temporal verification of reactive systems: safety", Springer Science & Business Media, 2012.

[5] D. Giannakopoulou, J. Magee, J. Kramer, "Fairness and priority in progress property analysis", Citeseer, 1999.

[6] E. M. Clarke Jr, O. Grumberg, D. Kroening, D. Peled, and H. Veith, *Model checking. MIT press: 2nd Edition*, 2018.

[7] E. M. Clarke, T. A. Henzinger, H. Veith, & R. Bloem, *Handbook of Model Checking, 1st ed.* Springer Publishing Co., Incorporated, 2018.

[8] E. M. Clarke, T. A. Henzinger, and H. Veith, "Introduction to model checking," in *Handbook of Model Checking.*, 2018, pp. 1–26. [Online].Available: https://doi.org/10.1007/978-3-319-10575-8_1

[9] E. M. Clarke, Jr., O. Grumberg, and D. A. Peled, *Model Checking*. Cambridge, MA, USA: MIT Press, 1999.

[10] C. Baier and J.-P. Katoen, *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

[11] J. Katoen, "Concepts, algorithms, and tools for model checking," *Arbeitsberichte des Instituts für Mathematische Maschinen und Datenverarbeitung*, vol. Band, no. 1, 1999.

[12] E. M. Clarke, E. A. Emerson, S. Jha, and A. P. Sistla, "Symmetry reductions in model checking," in *International Conference on Computer Aided Verification*. Springer, 1998, pp. 147–158.

[13] E. M. Clarke, R. Enders, T. Filkorn, and S. Jha, "Exploiting symmetry in temporal logic model checking," *Formal Methods in System Design*, vol. 9, no. 1-2, pp. 77–104, Aug 1996.

[14] E. A. Emerson, "Temporal and modal logic," in *Handbook of Theoretical Computer Science (Vol. B)*, J. van Leeuwen, Ed. Cambridge, MA, USA: MIT Press, 1990, pp. 995–1072. [Online]. Available:http://dl.acm.org/citation.cfm?id=114891.114907

[15] A. Valmari, "A stubborn attack on state explosion," *Formal Methods in System Design*, vol. 1, no. 4, pp. 297–322, Dec 1992. [Online]. Available: https://doi.org/10.1007/Bf00709154

[16] M. Vardi and P. Wolper, "An automata-theoretic approach to automatic program verification (preliminary report)," in *Proc. of 1st IEEE Symposium on Logic in Computer Science. IEEE*, 01 1986, pp. 332–344.

[17] O. Kupferman, M. Y. Vardi, and P. Wolper, "An automata-theoretic approach to branching-time model checking," *J. ACM*, vol. 47, no. 2, pp. 312–360, Mar. 2000. [Online]. Available: http://doi.acm.org/10.1145/333979.333987

[18] E. M. Clarke, E. A. Emerson, and A. P. Sistla, "Automatic verification of finite-state concurrent systems using temporal logic specifications," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 8, no. 2, pp. 244–263, 1986.

[19] R. Gerth, R. Kuiper, D. Peled, and W. Penczek, "A partial order approach to branching time logic model checking," *in Proceedings Third Israel Symposium on the Theory of Computing and Systems*, Jan 1995, pp. 130–139.

[20] F. Zambonelli, N. R. Jennings, and M. Wooldridge, "Developing multiagent systems: The Gaia methodology," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 12, no. 3, pp. 317–370, 2003.

[21] M. Wooldridge, N. R. Jennings, and D. Kinny, "The Gaia methodology for agent-oriented analysis and design," *Autonomous Agents and multi-agent systems*, vol. 3, no. 3, pp. 285–312, 2000.

[22] Kurt Jensen and Lars M. Kristensen. "Coloured Petri Nets: Modelling and Validation of Concurrent Systems". (1st ed.). Springer Publishing Company, Incorporated, ISBN 9783642002830, 2009.

[23] L. M. Kristensen, S. Christensen, K. Jensen, "The Practitioner's Guide to Coloured Petri Nets". *Int. J. Softw. Tools Technol. Transf.* 2(2), 98–132 1998.

[24] Kurt Jensen, Lars Michael Kristensen, Lisa Wells. "Coloured Petri Nets and CPN Tools for modeling and validation of concurrent systems" Published online: 13 March 2007 © Springer-Verlag 2007.

[25] Kurt Jensen, "Coloured Petri Nets. Basic concepts, analysis methods and practical use". Basic Concepts, vol. 1. Springer, Berlin 1992.

[26] Kurt Jensen, "Coloured Petri Nets. Basic concepts, analysis methods and practical use". Analysis Methods, vol. 2. Springer, Berlin 1994.

[27] Kurt Jensen, "Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use". Practical use, vol. 3. Springer, Berlin 1997.

[28] J. D. Ullman, "Elements of ML Programming". Prentice-Hall, Englewood Cliffs 1998.

[29] Standard ML of New Jersey. http://www.smlnj.org. [accessed at: 17 November 2019]

[30] J. Billington, G. E. Gallasch, B. Han, "A Coloured Petri Net approach to protocol verification". In: *Desel, J., Reisig, W., Rozenberg, G. (eds.) Lectures on Concurrency and PetriNets. Advances in Petri Nets. In: Proceedings of 4th Advanced Course on Petri Nets*, Lecture Notes in Computer Science, vol. 3018 pp. 210–290. Springer, Berlin 2004.

[31] J. Billington, M. Diaz, G. Rozenberg, (eds.), "Application of Petri Nets to Communication Networks", vol. 1605. Springer, Berlin 1999.

[32] Wolfgang Reisig, "Elements of Distributed Algorithms: modeling and analysis with Petri nets / Wolfgang Reisig". Springer, Berlin, New York, ISBN:3540627529 1998.

[33] M. A. Adamski, A. Karatkevich, M. Wegrzyn (eds.), "Design of Embedded Control Systems". Springer, Berlin 2005.

[34] Alan A. Desrochers, Robert Y. Al-Jaar, "Applications of Petri Nets in Manufacturing

Paper Titled: **Approach for the Formal Specification, Modelling, and Verification of a Safety-Critical Multi-agent System; An Employees Management Multi-agent System (EM-MAS)**

Certificate

The subject article has not been published or submitted or accepted for publication in any form, in any other journal or conference etc.
We also guarantee that the authorship of this article will *not* be contested by anyone whose name(s) is/are not listed by us here and we will not request the authorities of journal to add any more author(s) after its submission.

Signature of Corresponding Author

Authorship and Contribution Declaration

Sr.#	Author-s Full Name	Contribution to Paper	Signature
1	**Dr. Nadeem Akhtar** (Main/principal Author)	Proposed topic, basic study Design, methodology and manuscript writing	 Signature by the Corresponding author on Behalf of Co-Authors
2	**Dr. Maruf Pasha** (2nd Author)	Data Collection, statistical analysis and interpretation of results etc.	
3	**Mr. Abdul Rehman** (3rd Author)	Literature review & Referencing, and quality insurer	
.	.		