American Journal of Artificial Intelligence
2021; 5(1): 1-16
http://www.sciencepublishinggroup.com/j/ajai
doi: 10.11648/j.ajai.20210501.11
ISSN: 2639-9717 (Print); ISSN: 2639-9733 (Online)

An Intelligent Agent Model and a Simulation for a Given Task in a Specific Environment

Safa'a Yousef Abu Hadba, Ibtehal Nafea

Computer Science and Engineering College, Taibah University, Al-Mdinah Al-Monwara, Saudi Arabia

Email address:
safa_abu.hadba@yahoo.com (S. Y. A. Hadba), inafea@taibahu.edu.sa (I. Nafea)

To cite this article:
Safa'a Yousef Abu Hadba, Ibtehal Nafea. An Intelligent Agent Model and a Simulation for a Given Task in a Specific Environment. *American Journal of Artificial Intelligence*. Special Issue: *Emerging Trends in Artificial Intelligence with Machine Learning and Nature Inspired Algorithms*. Vol. 5, No. 1, 2021, pp. 1-16. doi: 10.11648/j.ajai.20210501.11

Received: November 11, 2020; **Accepted**: January 28, 2021; **Published**: February 10, 2021

Abstract: This paper introduces An Intelligent Agent Model for a Given Task in a Specified Environment. The methodology adopted in this work is based on mixing computational methods and functions to build an intelligent agent model. This paper focuses on building an intelligent agent model as a knowledge-based system that interacts with a dynamic environment for performing tasks. The class structure used to represent the environment in the knowledge base relies on three types of knowledge representation forms: production rule, semantic net, and frames. Each object in the environment is an instance of the class environment. Algorithms and functions are used to get knowledge from the state space of an environment to construct a task. The intelligent agent model can understand the environment from any position and can detect many subtasks, arrange them in a queue for execution, and can make decisions at a high scale of thinking. This model is proposed to maintain that an agent which is characterized by sufficiently low computational costs can interact with the environment in real-time but is powerful enough to reach the assigned goals in complex environments and within an acceptable time period. The intelligent agent model can calculate persistent changes in an external dynamic environment and any unexpected change, for example detecting the being of any problem in the environment and avoiding it. The intelligent agent can also learn and take reasonable decisions in the dynamic environment and automatically select an action based on task features. Thus, the intelligent agent can resolve several different kinds of difficulties.

Keywords: Intelligent Agent, Dynamic Environment, Performance

1. Introduction

An intelligent agent could be defined as a program that collects information or performs some other service without user present, sometimes called a 'bot' (shortcut for robot). The main abilities of intelligence are: acting, sensing, understanding, reasoning, and learning. Therefore, intelligent agents can resolve both easy and difficult problems, that need minimal knowledge engineering effort with perfect performance, compared with human-engineered intelligent agents. To outline an agent, we must define the task environment, type of environment, number of agents, and access the task in the environment [1].

There are two aspects in task specification that should be considered: the task environment space and the task action space. Thus, compatible with these two aspects, to specify and perform certain tasks, the next problems have been recognized:

1. How to explain the state space and action space in the environment; and
2. Identify and compile the required information and identify and design functions/algorithms (or utilize existing ones) required to practice the agent to perform the essential tasks.

The new development in the design, for an agent to become an intelligent agent, can be concrete in various environments to save money and effort and decrease work load. The intelligent agent can achieve many tasks respecting to the considerable environment, particularly in dealing with spatial data to support the security and emergency services overcome any problems in the environment and new mathematical equations to calculate whether intersection paths have problems or intersections, and ways to keep away from them.

These search techniques rely on tested information, which is established in the problem space. Each object in the

environment has an initial weight value, which changes with the transition of objects. Each object is related with the object-environment. Experiential search techniques are not related to any constraints and changes in the environment can provide a faster solution.

For an untrained agent, finding an object in any environment can be a very hard and time-consuming task. On the other hand, the same process for a trained professional may be quite a simple process.

The objective of developing a model of an intelligent agent in any environment, mainly in dealing with spatial data, is to help workers in such an environment to search and detect an object or complete a task.

Searching for a specific object in an environment needs much effort and commonly aspects several problems. There are some problem-solving methods used by humans that can be useful in this situation. Therefore, the implementation of these problem-solving methods, such as a software model, will be quite useful to people, especially those who working in risky environments.

This work is based on our previous work [10] which aims to utilize an intelligent agent that knows an environment and can carry out any action separately. The intelligent agent can interrelate with the environment and transaction between states without any user intervention, or in other words, unsupervised. Thus, the intelligent agent will work towards the specified aim by recalling the stored knowledge from the knowledge basis to complete that job. Moreover, the intelligent agent can interrelate with a dynamic environment and keep away from any obstacles. In this paper, we offer relevant works. Then we present the design's description of the proposed model of an intelligence agent as a knowledge-based system composed with the illustration of the knowledge basis, action space, and environment. After that, we show the design and implementation of the proposed model. Lastly, we conclude and state the future work.

2. Related Works

In recent years, the way that an intelligent agent interacts with an environment and completes a task requirement representation, has concerned numerous researchers in the area of intelligent agents and artificial intelligence.

Woolderidge et al. [2] declare that they use the results determined in several theories to analyze the difficulty of agent fulfillment for three classes of task specifications, achievement and maintenance tasks, and tasks specified as arbitrary Boolean set of achievement and maintenance tasks.

We explored how the features of an agent's units are affected since the agent must interact with an unidentified environment and, simultaneously, build an incremental and inductive model of it through symbolic learning. We need to maintain that an agent presenting adequately minimum computational budgets can interrelate with the environment in real-time however is strong enough to reach the allocated aims in complicated environments and within a suitable time.

Balduccini and Lanzarone [3] present an agent who independent interacts with an unidentified environment and constructs an incremental and inductive model of it. The important thoughts dealt with are:

1) Autonomy: The agent cannot depend on an oracle controlling its deductive processes.
2) Instrumentality: The agent should be capable to incessantly update the environment's model as stated by the knowledge resultant from interaction, exploiting as far as possible the information it has previously developed.
3) Inductivity: The agent should suppose rules based on detected examples, and it is, so, essential for it to continually verify the validity of the rules it has caused.
4) Unknown environment: The agent should complete well even once starting in a 'zero knowledge' state.

In the 1970s, it was lastly known that to produce a machine solve an intellectual problem, one had to know the solution.

Knowledge can be declared as a group of facts, events, procedures, and meta-knowledge. Typically, for a human to carry out a specific task, the distinct should recall the knowledge from his knowledge base, which is linked to that certain task.

The terms "procedural knowledge" and "path (sometime called route) knowledge" have distinct thoughts and are often misunderstood. The knowledge path has been defined by Niels [4] as, "A set of procedures that can be used only if an external entity explicitly specifies the initial situation and a desired end situation." It includes information about the arrangement of actions required to follow a specific path. A knowledge path is considered as knowledge around the actions to be achieved in the environment to effectively navigate paths between distant places, specially between a beginning and a destination.

Table 1. *Related works comparison.*

	Types of Environment	Number of Agents	Accessing Task in the Environment	Implementation of a Learning, Planning and First-principal Logic AI Techniques	Using an Agent Communication Language
(A)	1-Continuous 2-Dynamic	Single Intelligent Agent	1-Fully observable 2-Stochastic 3-Sequential	Yes	Yes Tell agent task about the write
(B)	Discrete	Single Intelligent Agent	1-Fully observable 2-Deterministic domain	No	No
(C)	Static Discrete	Single Intelligent Agent	Deterministic Sequential Fully	No	No
(D)	Dynamic Continuous	Multi	Sequential	No	No

Thorndyke and Goldin (1983) [5] describe knowledge kinds in the environment and refer to them as spatial

knowledge at three stages of information: landmark, procedural, and survey knowledge, for which each stage constructs on earlier stage.

Table 1 represents the comparison between four intelligent models A, B, C and D that are described in [6, 7, 11, 12]. The first row presents An Intelligent Agent Model and a Simulation for a Given Task in a Specific Environment [6]. Though, the second row demonstrates the implementation of an intelligent agent in a known, observable, discrete, and deterministic environment using a scriptable game-engine [7]. C is a static discrete environment using Crossword Puzzles in teaching [11]. The last row presents D which is agent-based traffic control and management systems [12].

3. Model Architecture

This part illustrates the methodology and the design of the proposed model. The methodology is the Software Development Life Cycle (SDLC) [9] which is divided into different phases with different organizations as the following:

1) Project Initiation & Planning

This phase is for describing the project's plan. It consists of the scope of the project, project organization, plans, estimated cost and risks management. In addition, it gives an estimation about the management plans of each thing in the project including resources, time budget and risks.

2) Analysis: This phase is for describing the requirements that are needed to develop the system. It includes the functional and non-functional requirements. Also, use cases that show the interactions between the user and the system and the requirements analysis. In addition, the databases design of the system. Moreover, the requirements needed for the project are going to be described and organized. Also, analysis of the system, and a description of the process model.
3) Design: This phase is for describing the design of the system. It involves internal and external user interfaces, a brief overview of the code and relationships between the objects. In addition, a full design presentation of the project with every necessary detail related to the system.
4) Implementation: This phase shows the real implementation of the system in terms of making the system ready to be used by the users.
5) Testing: This phase shows the testing of the whole project after implementation with all the tools, and methodologies used.
6) Maintenance: This phase is to ensure that the system is working smoothly without any problem. Also, support for the user is presented in this phase.

For design model it is built on the functional model of the human system as a knowledge-based system. It contains the definition of knowledge representation and how it is used in the proposed model. It similarly contains knowledge representation for the environment and state space and action composed with a meaning of the main algorithms as processes in a knowledge-based system. Once these three parts are ready, the modules are integrated into their processes, i.e., each module has a task that accompaniments other modules' tasks.

The construction of a knowledge-based system relies on the proposed functional model of the human system, which was made according to the route of the vertical top-to-bottom arrow to the left of Figure 1 [8, 10], as understood from relation with the environment unit, human inference engine, and long-term memory.

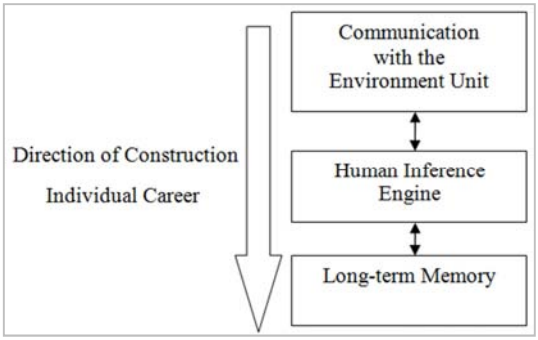

Figure 1. *Functional Model of human System.*

The knowledge-based system can be represented by the functional model of the human system but will be implemented as shown in Figure 2 from the bottom-to-top direction seen by the vertical arrow on the left side. Therefore, relation with the environment unit will be as the user interface. The inference engine will be human but will be beheld to characterize untouchable entities, which are; willingness, needs, and vision, Incentives, hobbies, and the outcome of the environment as a problem-solving method, search technique, and reasoning agent. Lastly, the long-term memory will be the knowledge base. The most significant section of the knowledge-based system is the knowledge base itself, then all the other sections of the system depend on its implementation.

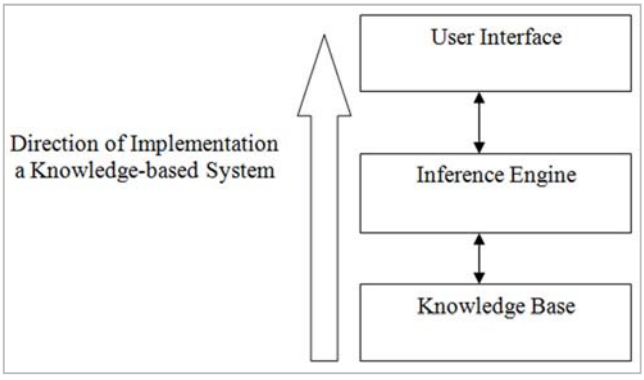

Figure 2. *Structure of the Knowledge-based System Functional Model of Human System.*

Figure 1 shows the structure of the human functional system and Figure 2 shows the implementation of the knowledge-based system as a simulation of the functional model of the human system.

Figure 3 shows the intelligent agent as the knowledge-based system of using state space and action space for an intelligent agent model.

The proposed model is built on the common construction of

the knowledge-based system as shown in Figure 2 and contains of four modules: user interface, inference engine, knowledge base, and working memory. Utilizing the proposed model for searching for an object in an environment requires the accomplishment of many tasks.

The intelligent model should be able to detect and analyze the path required for a task. This capability uses the intelligent agent to extract the needed data and information as knowledge, giving to the common construction of a knowledge-based system.

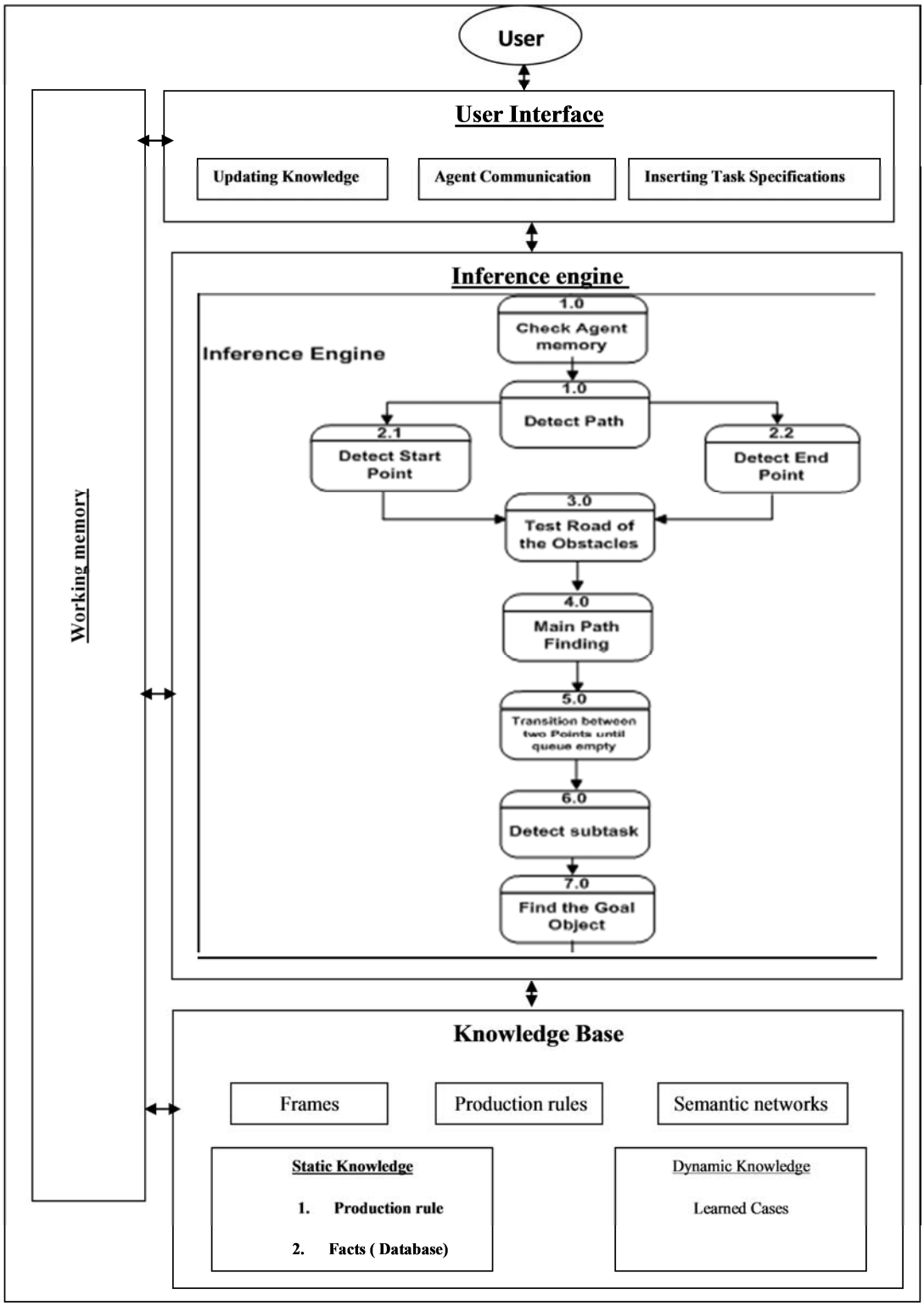

Figure 3. *Proposed model of intelligent agent as a knowledge-based system.*

A. User Interface Module

The user interface simulates the communication with the environment unit of the functional model of the human system. Therefore, this module contains several functions; all

functions allow the user to interact with the knowledge-based system. When a user utilizes the intelligent agent, he should first enter a password and a user name, which allows interaction with the agent. The user interface contains three modules as showed in Figure 3:

1) Inserting task specifications: can insert the task within part of the proposed model, for example, bring book by ID, bring book by name, replace two books, and return book to bookcase.
2) Agent communication: this relays on the interaction with the environment. The intelligent agent model can estimate the persistent changes in the external dynamic environment and any unexpected change, like detecting the presence of any obstacle in the environment and avoiding it.
3) Updating knowledge: when changing the stored knowledge and deleting it, and insertions of new knowledge are applied according to the assumed request by the end user.

B. Interface Engine

The inference engine includes three modules: problem solving method, search technique, and reasoning agent. Their main functions are calling the right knowledge from knowledge base, when required, to solve a problem, according to the given statements, and applying a heuristic search technique. And simplify the ambiguity in the new developed knowledge inserted in the working memory, by calling an existing knowledge from the static knowledge base that is added to the dynamic knowledge base. For instance, while change happens in the external dynamic environment, such as observing the existence of an obstacle in the environment and avoiding it, the intelligent agent recalls the correct knowledge from the table and uses a heuristic search technique to find a quick solution.

The reasoning agent is usually one of many techniques, such as forward chaining, backward chaining, or mixing both. In this approach, according to the circumstances of the provided goal to the proposed intelligent agent, we will have used both.

The algorithms and function will be used by the inference engine of the intelligent agent. Moreover, these algorithms are used for obtaining knowledge from the problem's state space, the environment's problem space, and the action's state space to complete a specific task.

When we sort objects' ID, the first phase is the intelligent agent search for the availability of the object (book) matching to the experience (in this case, dynamic knowledge). The intelligent agent then goes to the next phase to finish the task, finds the start and end points, tests the road of the obstacles, afterward transits between two points and detects a subtask.

C. Working Memory

The working memory is defined as the place where all inference engine, user interface and knowledge activities must be carried out. These activities are:

1) The application of the reasoning agent used: the reasoning agent can normally be one of several techniques, for example forward chaining, backward chaining, or both. In the proposed model, both are used corresponding to the state of the assumed goal and the proposed intelligent agent;
2) The application of searching techniques used: applying a heuristic search technique to refine the ambiguity in the new acquired knowledge inserted in the working memory; and
3) The processes of agent interaction within the environment: the intelligent agent can learn and take reasonable decisions in the dynamic environment and regarding any unexpected changes, such as detecting the being of an obstacle in the environment and avoiding it.

D. Knowledge Base

The knowledge base characterizes the repository of knowledge for a restricted and specified domain. The knowledge could be outlined as a group of facts, rules, events, and meta-knowledge. Generally, knowledge could be either declarative or procedural. In this paper, both declaratives are consumed: dynamic and procedural, or static, knowledge.

The basic structure of static knowledge is a knowledge base that includes domain knowledge used for problem solving. The formula of a rule can be modified in Horn Clause formula:

$$\text{Action} \leftarrow \text{condition1}, \text{condition2}, \ldots \text{conditioni}$$

$$\text{Where i} \geq 0 \text{ and is an integer number.}$$

Dynamic knowledge includes knowledge developed through the run time of the system using the interaction of the user with the proposed system. This knowledge provides cases for specific problems or facts. The basic structure of dynamic knowledge is the database that contains a structure of facts used to be compatible with the (condition) part of instructions stored in the knowledge base.

Bear in mind, in this work, frames in incorporation with production rules and semantic networks are all manipulated to characterize knowledge in the knowledge base.

In 1974, Marvin Minsky proposed the meaning of frames as structures in which each frame had its own name and a set of attributes, or slots, associated with it. Wesley (2005) described why is it was essential to use frames and provided an accepted method of structured representation of knowledge. In a single entity, a frame links all essential knowledge regarding a specific object or perception. The frame prepares a means of organizing knowledge in slots to explain different attributes and features of an object.

A semantic network is a directed graph, containing of nodes. Every node is an object in the environment, and links (arcs) between objects: a link is the relationship between objects.

Figure 4 clarifies the semantic network for the perception environment that is applied for indicating knowledge related with the environment.

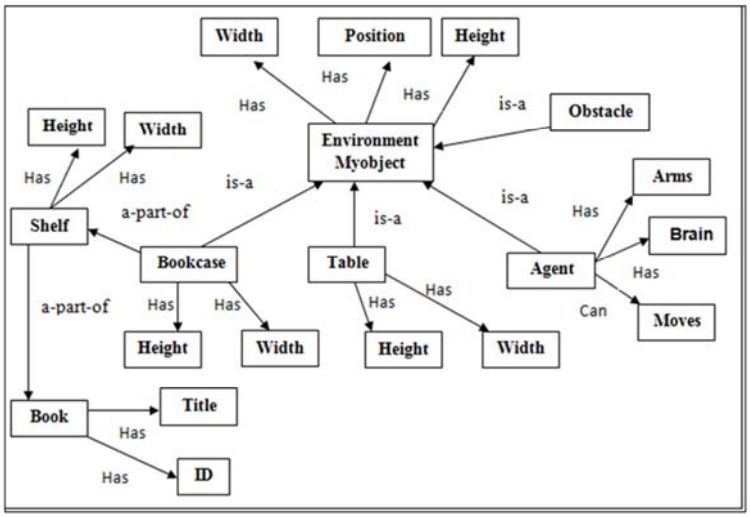

Figure 4. *Semantic networks showing some object concepts and properties.*

The rule base is a set of the form IF (conditions) THEN (Action) rules. The conditions are a part of a rule that contains zero or more conditions; the special case of zero occurs in case of a fact. The action part is the goal. Figure 5 illustrates an example of applying the production system using the working memory.

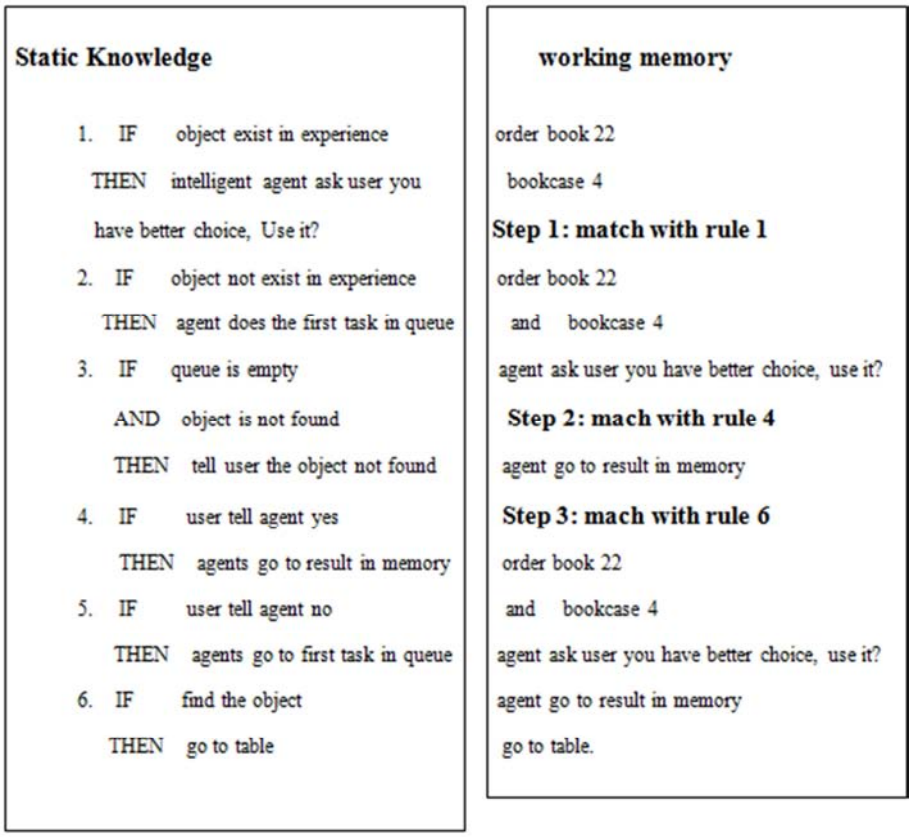

Figure 5. *Production system using working memory.*

For each cycle of the production system, the following actions will happen:
1) Selection: select the rules from the rule base, according to the goal.
2) Matching: match the selected rules according to the assertions given in the context (database).
3) Confliction: choose one of the matched rules according to the time and space.
4) Fire (execute) the chosen one.
The environment has several objects and every object combinates another object in the environment; every object denoted by

the class includes the properties and methods that define the behavior of the object, in which properties describe the distance and class of the object or its blueprint. Figure 6 illustrates the hierarchical structure of the environment.

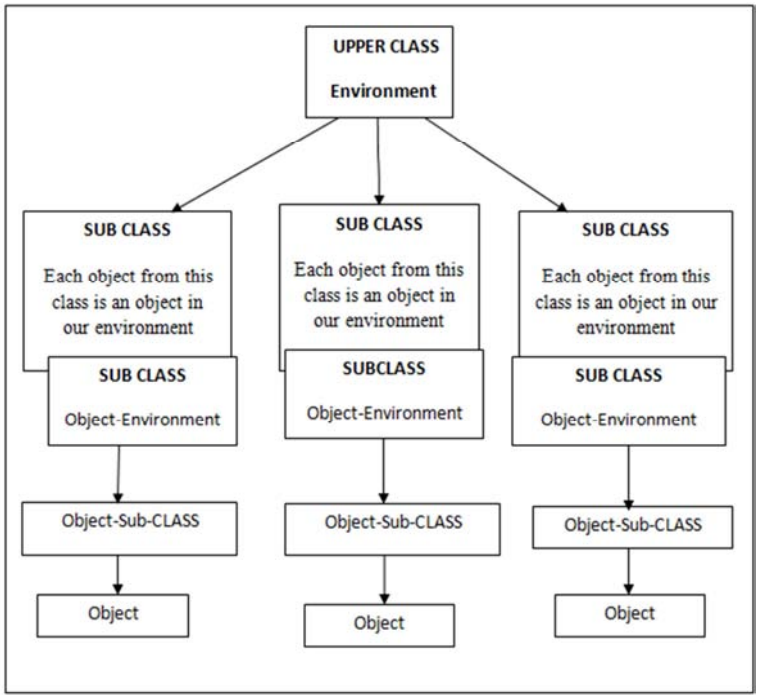

Figure 6. *Hierarchical Structure of Environment.*

Figure 7 illustrates the hierarchical structure of the environment for the proposed model library and presents the object that includes the library. For instance, the bookcase and the table and the composite of objects, for example bookcase combined shelf and shelf combined book; each object includes properties and measurements commensurate with the item having this object, which also fits with the environment and the location of each object, depending on the design environment.

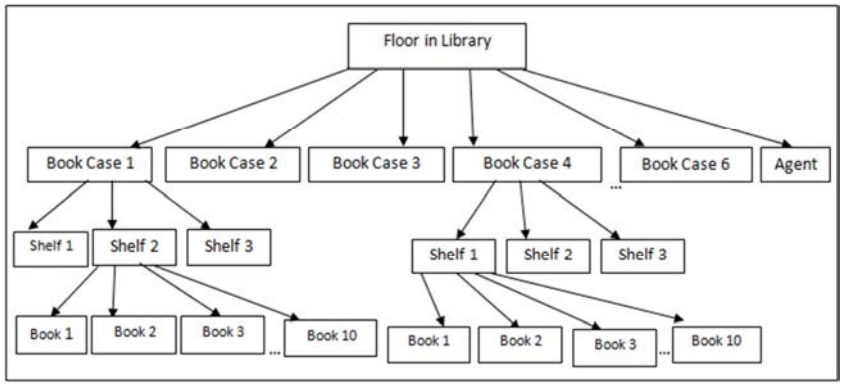

Figure 7. *Hierarchical Structure of the Proposed Model (Library).*

4. Case Study

In the proposed case of the library environment, the spaces will be distributed into two kinds: how to represent environment space (state space) and how to characterize (action space) an event. Thus, the first step of doing this characterizes the environment.

State Space is an entire and comprehensive explanation of the environment that the intelligent agent will perform within, containing:

1) Physical space layout example: building/space measurements;
2) Inner spacing layout example: shelf details;
3) Constraints example: relative allowed distance;
4) Obstacles (permanent or temporary) example: columns/sculptures, or maintenance.
5) Limitations example: moving objects.

Action Space is a complete list of available actions and events within the given environment. These can be categorized into two main groups:

General/simple actions:

Examples: MoveForward ( ), MoveBackward ( ), MoveLeft ( ), MoveRight ( ).

For the intelligent agent in the environment, to get the commanded task, assume that the environment is collected of a set of finite states, Set S={s_1, s_2…}.

The intelligent agent should transfer from one state to another when given the transition task of producing an event in which the environment contains a set of finite occurrences.

Set E={e_1, e_2, …}.

The main task in this environment will contain a subtask sequence of states and events. Main Task t: $S_1 \rightarrow^{e1} S_2 \rightarrow^{e2} S_3 \rightarrow^{e3} S_4 \rightarrow^{e4} ... \rightarrow^{en-1} S_n$

Let t be a set of all such possible tasks and the transition between two states contain the event: TR: $2S \rightarrow e_1$

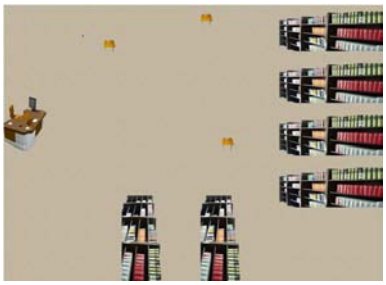

Figure 8. *Environment Proposed model.*

Figure 8 illustrates the environment-proposed model, which contains a single intelligent agent. The intelligent agent could interact with the environment and so perform the task; moreover, intelligent agent transitions between states until the task is complete.

S is a set of states and E is a set of events performed by a agent in the environment; t is a task done by the agent and the task contains many transitions among the state and doing the event in each of the two states. The s1 is the current state to agent.

The environment contains of a set of states and events, the transitions and the initial state, and intelligent agent. Environment={S, E, A, t, s1}.

An example of the state transition of an environment is represented in Figure 9; the arcs among two states demonstrate the sets of actions corresponding to transitions.

Every state in the graph is the object of a class and the state S2, S3, S4, S5, S6, and S7 are the bookcase in a proposed model; the state S1 is a table in the library and S0 is the initial state mean, the current position of agent.

The agent's moves are detected based on the task given, such as the user need to bring a book, exchange it between two books, or return it. When the agent is given the task, he can then interact with the environment until he reaches the goal.

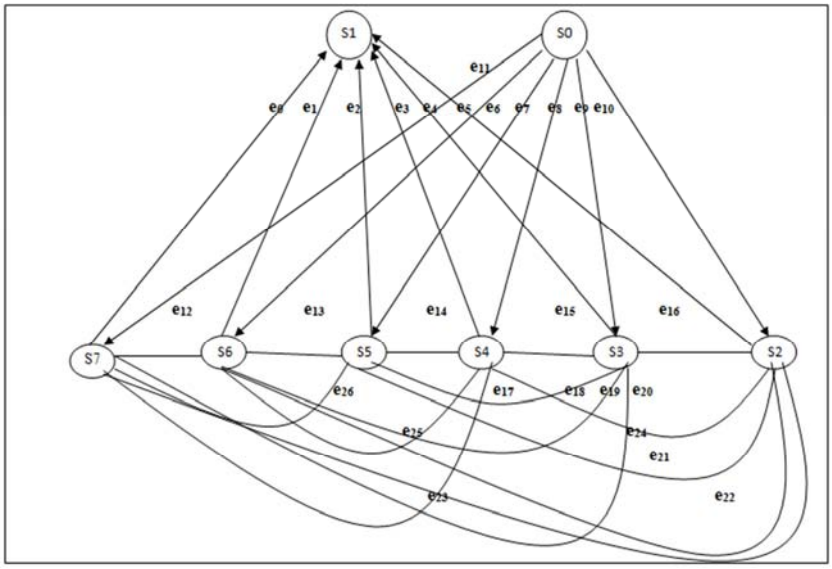

Figure 9. *Transition in the Environment (state, event).*

The environment state transition is presented by the graph in Figure 9. In this environment, an intelligent agent controls many available actions, and the intelligent agent in the environment is not allowed to execute the same action twice. Arcs between the states in Figure 8 show the actions that affect the state transitions.

Note: the start position of the intelligent agent state S0 for each goal.

1) Goal1={S3}

An intelligent agent can constantly reach goal1 by performing an action in the start position of the intelligent agent, choosing an action e10, e9, e8, e7, e6, or e11; the result of which will be either S2, S3, S4, S5, S6, or S7. If S5 results, the intelligent agent can perform e13, e26, e21, e18, e2, or e14 the result of which will be either S2, S3, S4, S6, S1, or S7. The S3 result can simply perform e18.

2) Goal2={S3}

An intelligent agent can reliably achieve goal2 by the performing action in the start position of intelligent agent choice of the actions e10, e9, e8, e7, e6, or e11, the result of which will be either S2, S3, S4, S5, S6, or S7. If S2 is a result, the intelligent agent can perform e22, e24, e21, e16, e5, or e23, the result of which will be either S5, S3, S4, S6, S1 or S7. If S1 is a result, it can simply perform e5, which is not allowed. Mean does not allow it to go to the table before it finds the book.

Declare the main process, in memory of the intelligent agent; when given the task, the first step is to determine the task and determine how to reach that goal. The next step is then to analyze the knowledge in the environment state space and action space, which are related to the goal.

The intelligent agent is able to recognize the surrounding environment from any perspective and identify the necessary subtasks, organizing them into a sequence, then execute them. The start and end point based on the position of the intelligent agent for each subtask is located then the path is planned and the road is tested for obstacles, to determine the procedure needed to reach that goal. Finally, the task representation is stored in the knowledge base. Figure 9 shows the control data flow diagram for the task when ordering the Bring Book ID or Bring Book by Name.

Since the primary task of the intelligent agent model is to find and get an object, the user first inserts the object's characteristics. In the case of the library environment, the object's features could feature many aspects, e.g. as the name of the book, its number, and place. Resultantly, the intelligent agent seeks the object's accessibility in relation to the experience, i.e. in this instance, dynamic knowledge. If it is present, the intelligent agent moves to the next phase in order to complete the job. If not present, the intelligent agent instigates the process 'fillIndexQueue', executing the job in a specific order to achieve the target within the environment.

When ordering an object's ID, the first step is the intelligent agent's search for the availability, in relation to the experience, i.e. in this instance, dynamic knowledge. If it is present, the intelligent agent moves to the next phase in order to complete the job. The intelligent agent understands the book's path within the stored knowledge base due to prior experience.

When taking another action, for example, replacing books, the agent brings the data up to date by utilizing dynamic knowledge. Learning from experience, the agent selects the most appropriate response at the right time. In addition, an agent can interplay within the environment with no support from the user or guidance for making decisions. In Figure 10, the state transitions are exhibited, demonstrating the progression for start-up and presenting processing and intelligent behavior interactions.

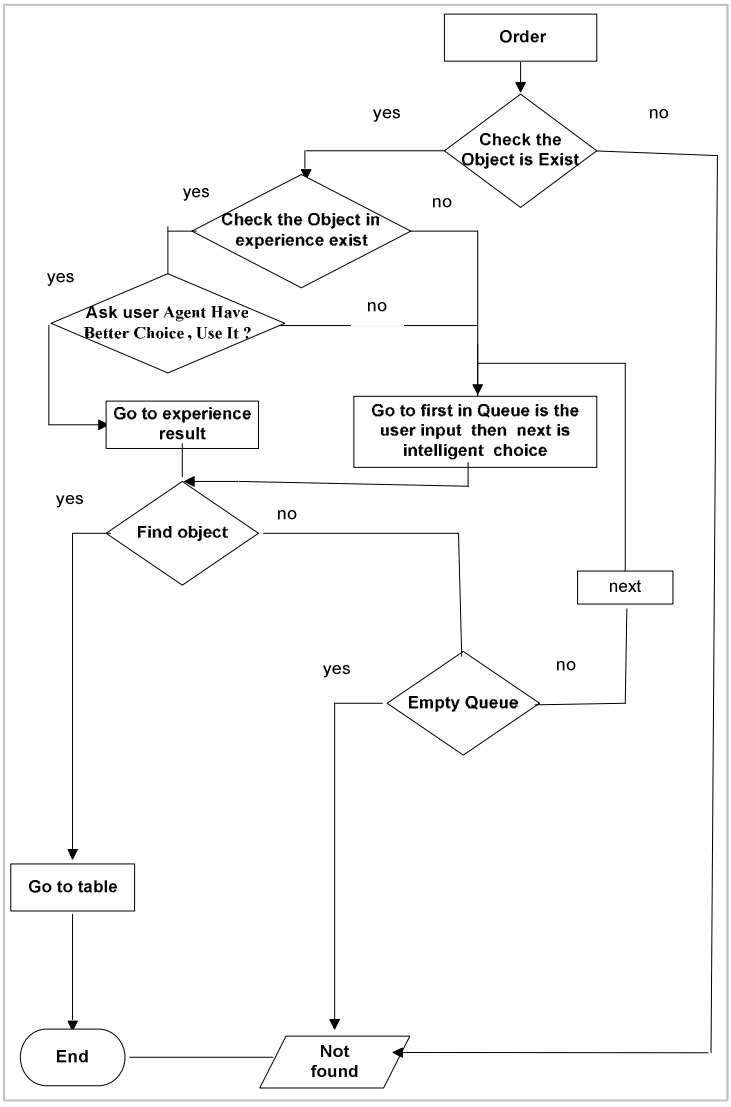

Figure 10. *The Control Data flow diagram for the task when ordering Bring Book ID.*

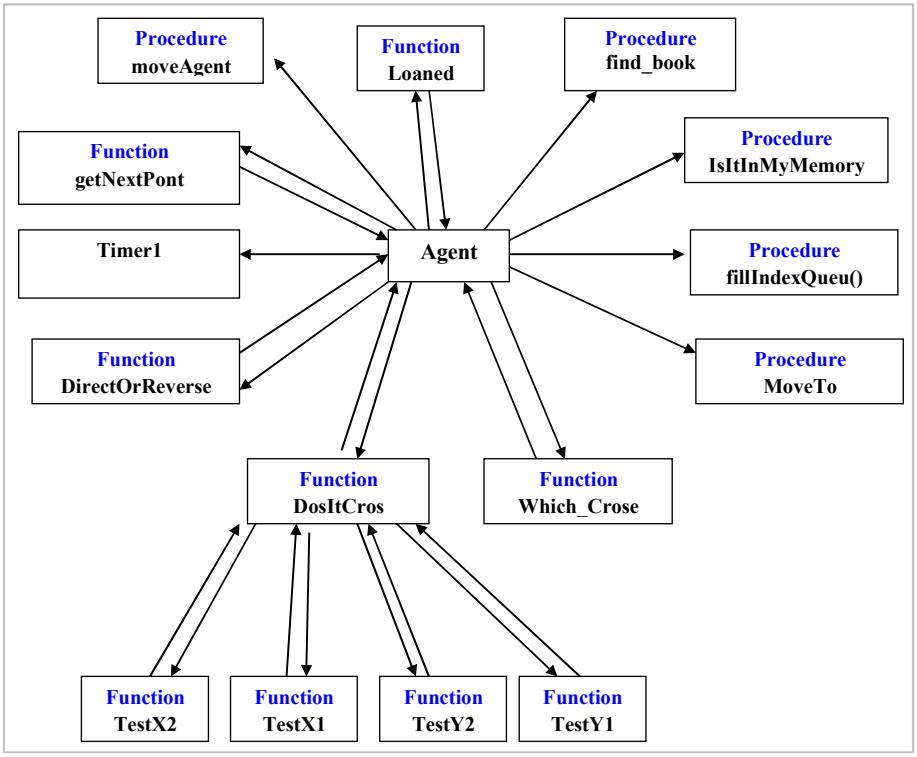

Figure 11. *A diagram of the states' interaction with transitions.*

To inform the agent about the requirements of the user the detection of the task is necessary, the agent then begins interacting within the environment. The initial function is the loan check to see if the book has been loaned out or not. If it has not been loaned the next procedure is to find the book. It must first be checked if book in dynamic knowledge mean in experience to agent by the procedure Is It In My Memory if agent exists book in experience then go to the procedure Fill Index Queue and put the first number in queue number bookcase exist object in dynamic knowledge else if it does not exist in experience then continue and fill the queue intelligently.

The main task can then be carried out: execute the sub-task in queue, procedure "Move To", find the bookcase it needs to go to, then carry out the action "Which_Crose", and check if any obstacles intersect with the path when going to the bookcase.

Function Dos It Cros has four functions: Function TestX1, Function TestX2, Function TestY1, and Function TestY2. Each calculates whether the intersection path has an obstacle or no intersection.

The Direct Or Reverse function checks any direction go agent for the most efficient time and the shortest possible path.

Timer1 executes the walk: each step is 0.01 seconds and then it calculates the next point in the line until the walk arrives at the end point; use the function get Next Pont to calculate the next point in line, then move to the next point until the end point via the procedure Move Agent. On arrival at the first target, the agent carries out a subtask to find the required object and match it to another object on the bookcase, then find the object and go to the table or continue to execute the next task in the queue.

The structure of the class is utilized to demonstrate the environment within the knowledge base. This is reliant on three kinds of knowledge representations; the production rule and semantic frames and net in which every object is an instance of a class containing the following attributes and methods of the object within the environment:

1) Attributes: Things exist in the object and cannot be separated because of the object's properties and are concomitant with the object.
   a) The object position: each object contains many properties, describes the properties in the class, and each object has a position in environment.
   b) The length and width of the object: this attributes for each object in the environment, consisting of the dimension's length and width.
   c) Name: each object has a name.
2) Methods:

Methods of the object: what we can do to the object, anything that could be implemented by the object, such as intelligent agent, can move in the environment between the two states.

An object can be used to denote anything. It could be a concrete concept such as a book, a table or a bookcase. A class provides a framework for characterizing a type of object.

The class is the blueprint. An object is an instantiation of a class. For instance, a bookcase has features, such as the number of shelves or ID number. The class 'bookcase' provides a generic description using features, but an instance is an object that has values assigned to these features. The word 'property' is used to denote an attribute of an object.

Properties are pieces of information about an object, such as color, size, or number of shelves. Methods describe behavior because they state what an object can do, for example, an intelligent agent object that might have been moved describes the task content; the number of moves is a property. Figure 11 shows the Inheritance Class Structure for the whole object in the environment.

The class represents the object in the environment; each class concerts the blueprint of an object in the environment.

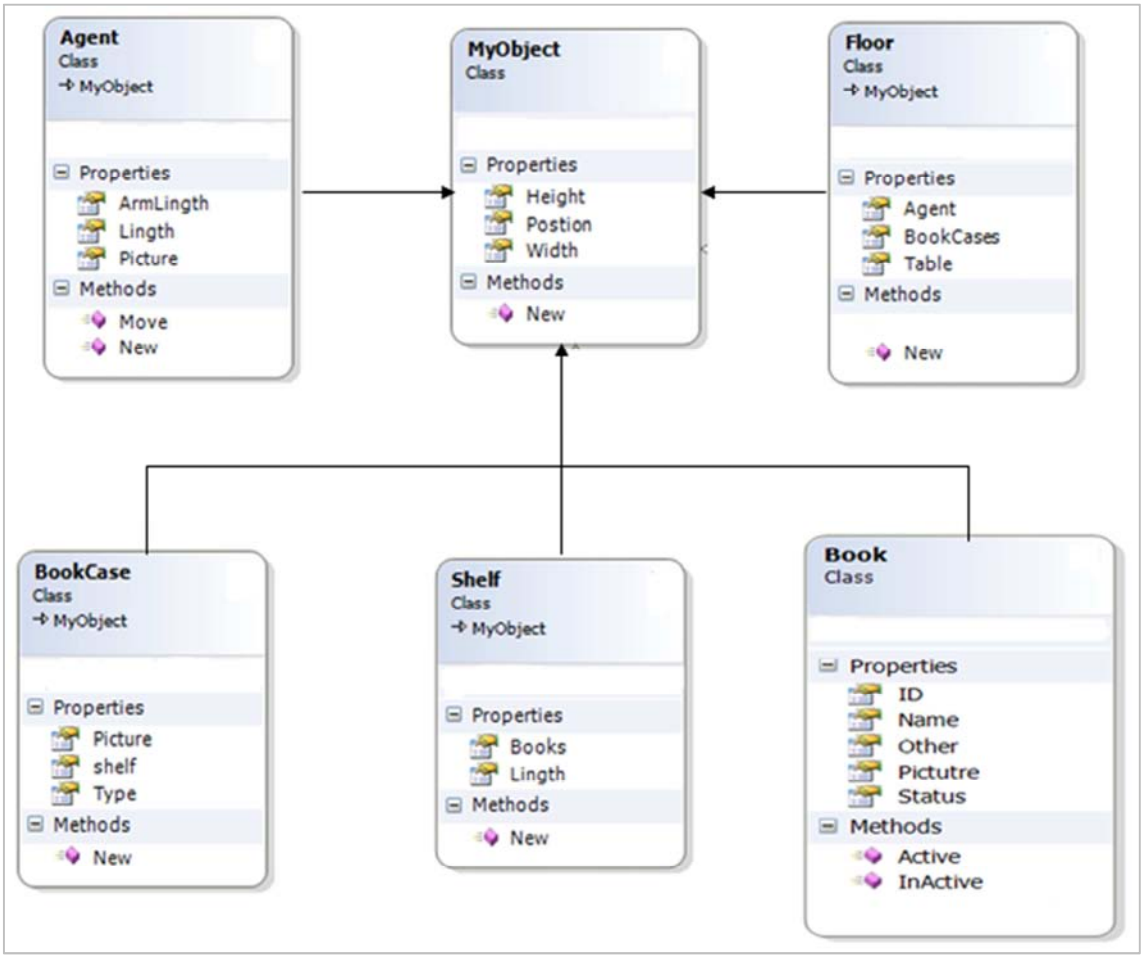

Figure 12. *Inheritance Frame Structure.*

Figure 12 presents the main algorithms as processes for the proposed model as a knowledge-based system for one process in the queue, which mimics human behavior (a librarian in this implementation) to recognize an object, in this case, a book in the environment (library) using the following steps:

The first step is a full description of the existing environment through the knowledge base, which will be described as state space and action space, and the existing storage environment.

The second step is to allow the intelligent agent model to understand and analyze the nature of the environment through the interaction of the user with their existing knowledge base, so we can plan the movement and complete the task of reaching the desired book.

The third step represents the path that draws the action of event and reaches the goal according to the book in the requirement

The functions and algorithms are utilized by the intelligent agent's inference engine. They are also utilized to attain knowledge deriving from the state space of the problem, the problem space of the environment, and particular tasks are achieved by the state space of the actions; these algorithms will be explained in this section.

Since the primary task of the intelligent agent model is to find and retrieve an object, s the user first inserts the object's characteristics. In the case of the library environment, the characteristics of an object has several aspects, e.g. the name of the book, its number, and where it is placed. Resultantly, the intelligent agent looks for the book's availability in relation to its working memory and dynamic knowledge. If it is present, the intelligent agent moves to the next phase in achieving the job. If it is not present, the intelligent agent initiates the Fill Index Queue procedure.

The procedure Fill Index Queue is used to arrange all the sub-tasks of the main one to be performed by the intelligent agent as seen in Figure 13.

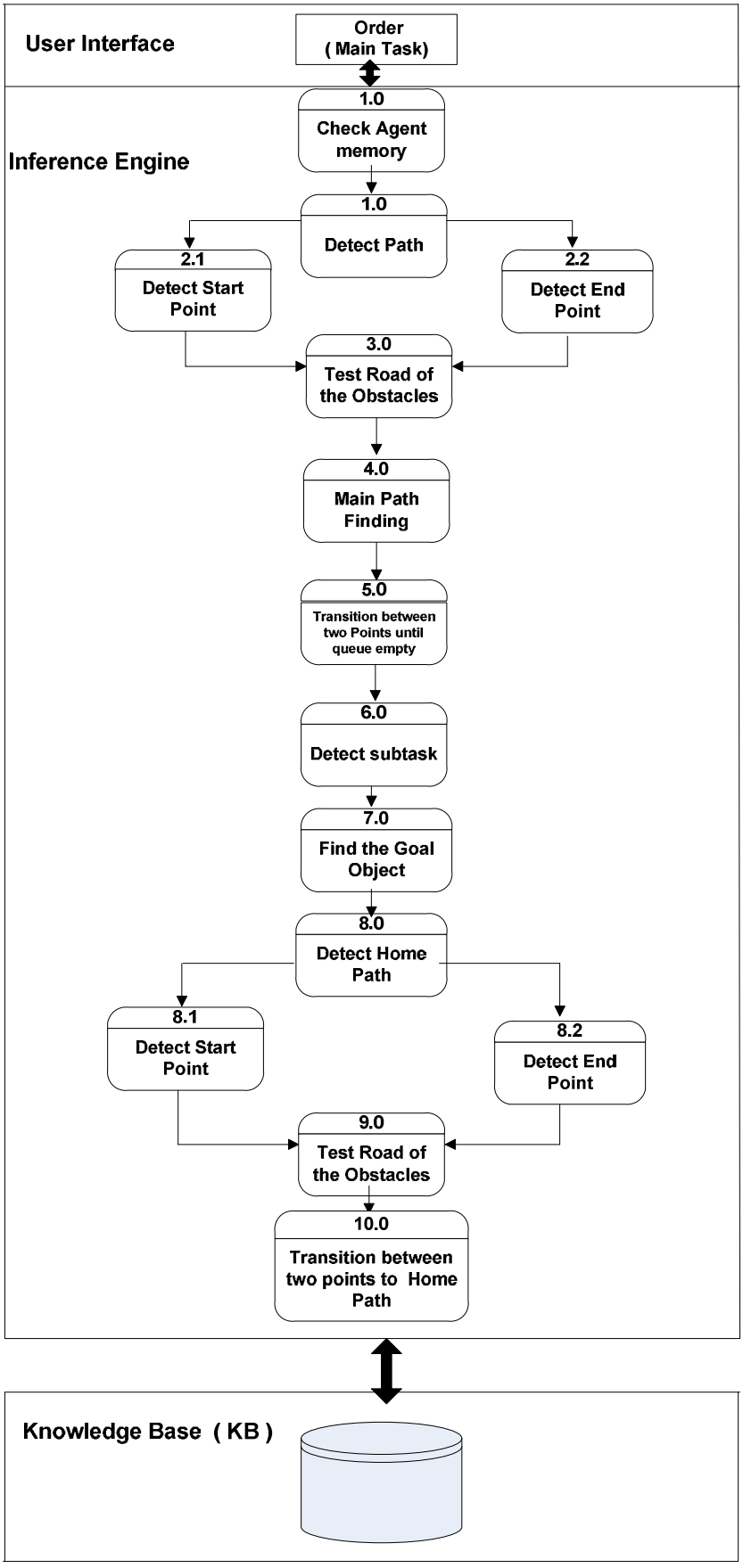

Figure 13. *Primary Algorithms as a Series of Actions within the Knowledge-based System.*

```
Sub fillIndexQueu(ByVal BID As String)

IndexQueu = GenNewQueu(BID, BookCaseIndex)

  End Sub
```

Figure 14. *Procedure Fill Index Queue.*

The inquiring search method is carried out as the function Gen New Queue. Its task is to determine a quicker, but not necessarily optimal, solution. These techniques are often reliant on acquiring knowledge, which is found within the problem space. Therefore, this function will depend on the weight of an object in its environment. Using initialization weight, it sorts all the weights to each object in all positions in the environment, as seen in Figure 14, so the general structure of the heuristic function is:

$$F(n)=g(n)+h(n)$$

g (n)=the search technique returns the weight of an object in the sub-environment

h (n)=the heuristic information for the object n

```
Function GenNewQueu(ByVal bid, ByVal indexToStart) As Queue
        Dim ds As New Data.DataSet
        ds.ReadXml("data.xml")
        Dim Q As New Queue
        Dim ar(5) As BBKS
        For i As Integer = 0 To 5
            ar(i).Vlaue = ds.Tables(0).Rows(i)("b" & bid)
            ar(i).Index = i
        Next
        ar = SortIt(ar)
        Dim ar2(4) As BBKS
        For k As Integer = 0 To 5
            If ar(k).Index = indexToStart Then
                Dim temp As BBKS = ar(0)
                ar(0) = ar(k)
                ar(k) = temp
            End If
        Next
        For o As Integer = 1 To 5
            ar2(o - 1) = ar(o)
        Next
        ar2 = SortIt(ar2)

        For u As Integer = 1 To 5
            ar(u) = ar2(u - 1)
        Next
        For j As Integer = 0 To 5
            Q.Enqueue(ar(j).Index)
        Next
        Return Q
    End Function
```

Figure 15. *Function for the Heuristic search Technique.*

```
Sub incresWeight(ByVal bid)

 Dim ds As New Data.DataSet
 ds.ReadXml("data.xml")
 h=1
 Dim g As Integer = ds.Tables(0).Rows(BookCaseIndex)("b" & bid)
 F = g + h
 ds.Tables(0).Rows(BookCaseIndex)("b" & bid) = F
       ds.WriteXml("data.xml")

End Sub
```

Figure 16. *Heuristic Function.*

Within the enviornment, every object has a weight value and this initial value alters due to object transitions; these objects are all linked to the object-environment, as seen in Figure 15.

In Table 2, the two-dimension array is explained. This comprises of the objects' transition weights. When it is required to occupy the whole of the queue, the bookcase must be rearranged into required visits, depending on the maximum weight in each column with the contained object needed.

Table 2. *The two dimensions array.*

Bookcase/object	**Object$_1$**	**Object$_2$**	**Object$_3$**	…	…	…	…	…	**Object$_n$**
Bookcase$_1$	Weight$_{11}$	Weight$_{12}$	Weight$_{13}$						Weight$_{1n}$
Bookcase$_2$	Weight$_{21}$	Weight$_{22}$	Weight$_{23}$						Weight$_{2n}$
Bookcase$_3$	Weight$_{31}$	Weight$_{32}$	Weight$_{33}$						Weight$_{3n}$
Bookcase$_4$	Weight$_{41}$	Weight$_{42}$	Weight$_{43}$						Weight$_{4n}$
Bookcase$_5$	Weight$_{51}$	Weight$_{52}$	Weight$_{53}$						Weight$_{5n}$
Bookcase$_6$	Weight$_{61}$	Weight$_{62}$	Weight$_{63}$						Weight$_{6n}$

Using a sorting algorithm is efficient for small arrays and it is used to sort the weight of each object in all sub environments. Therefore, this function insertion sorts as seen in Figure 16.

```
For j←1 to length (b) -1
num ← b [ j ]
i ← j - 1
while i >= 0  and  b [ i ] < num
   b [i+1] ← b [ i ]
i ←i-1
b [i+1] ← num
```

Figure 17. *The pseudo code for sorting insertion.*

As mentioned in the scenario, the intelligent agent should specify the start and end points. To recognize a start point, an algorithm is made. As shown in Figure 17, the start point is the initial state (the intelligent agent's starting position) for the transition between the initial state changes when moving from one state to another to perform a specific task, the end point of the $task_1$ is the starting point of the $task_2$, and so on, in each transition.

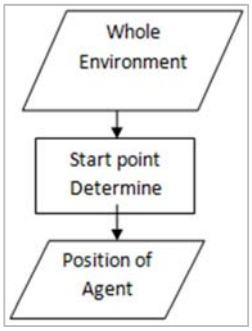

Figure 18. *Start Point Finding Algorithm Pseudo code.*

This algorithm is similar to the previous one, which was used to recognize the end point. An algorithm was made (as shown in Figure 18) to find the end point in the final state (position of target) for the transition between states. The end state changes when moving from one state to another to perform a specific task. The start point of the $task_2$ is the end point of the $task_1$, and so on in each transition

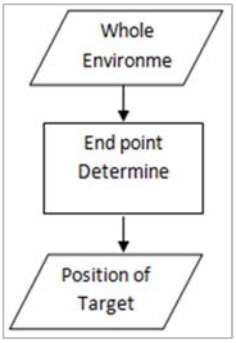

Figure 19. *End Point Finding Algorithm Pseudo code.*

As in the scenario, the start point should be in the main path of the targeted object. All of its pixels are then apart from the main path pixels.

As shown in Figure 19, to find the main path, the algorithm needs to get to the next point from the line between the start point and the end point.

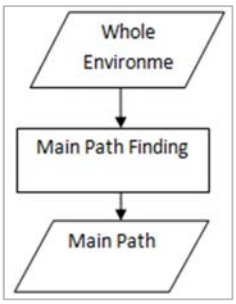

Figure 20. *Main Path Finding Algorithm Pseudo code.*

The Intelligent Agent finding the shortest path-finding algorithm used in this thesis needs to detect the minimum distance between two points, using a well-known mathematical equation of distance between two points:

$$D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Where:

D: The distance between two points

(x_1, y_1): The start point

(x_2, y_2): The Goal point

Trying to obtain the shortest path, the algorithm will calculate the minimum distance between (goal point) and all pixels through the path starting from the (start point). The algorithm initially takes the first location (start point) and finds the distance between it and the (goal point), then it calculates the next point in line of the current point (start point). The algorithm will continually calculate the pixels one-by-one and choose the pixel that satisfies the least distance between the selected pixel and the (goal point) The algorithm will repeat this process on each line in the main path between the two states until the last point of the (goal point).

The function explains the next step to the intelligent agent, which must move to reach the target as shown in Figure 20.

```
'Function calculate the next point in line between two state

Function getNextPont(ByVal _start As Point, ByVal _end As Point, ByVal
index As Double, ByVal L As Integer) As Point
        Dim x As Integer = (_start.X) + index * (_end.X - _start.X)
        Dim y As Integer = (_start.Y) + index * (_end.Y - _start.Y)
        Return New Point(x, y)
    End Function
```

Figure 21. *The function 'get Next Point'.*

If the next point in line is calculated from the current point (start point) and the algorithm will continually calculate the pixels one-by-one and choose the pixel that satisfies the least distance between the selected pixel and the (goal point). The algorithm will repeat this process on each line in the main path between the two states until the last point of the (goal point). Index is the step of the agent divided by the length of the distance between the start point and the end point.

Index=1/ lingth

$$\text{Length}=\sqrt{(s.x-e.x)^2+(s.y-e.y)^2}$$

X=start.x + index * (end.x – start.x)
Y=start.y + index * (end.y – start.y)
Next point (X, Y)

Find next point (x_1, y_1) to the Current point, in line between two states. Each step to the agent is executed in 0.01 seconds; the next calculate state each transition to next state when count final equal one then line finish then go to next line or if not contain next line mean that is current position is a target.

The execution of the main task can start and the execution of the sub-task is in the queue. Use the procedure MoveTo, detect the bookcase that it needs to go to, then carry out Which_Crose to analyze whether there are any obstacles intersecting with the planned path on its way to the bookcase.

DosItCros calls for four functions: Function TestX1, Function TestX2, Function TestY1 and Function TestY2. These are calculated if the intersection path has obstacles but not an intersection.

To achieve the significance of X1, X2, Y1, and Y2 the subsequent functions along with equation calculations can be checked at any intersection point of the obstacles that are present within the environment.

$$\left.\begin{aligned} Index &= \frac{(obstacle\ location.x - Start\ position.x)}{(End\ position.x - Start\ position.x)} \\ Y1 &= Start\ position.y + Index * (End\ position.y - Start\ position.y) \end{aligned}\right\} \quad (1)$$

$$\left.\begin{aligned} Index &= \frac{(obstacle\ location.x + obstacle.width) - Start\ position.x}{(End\ position.x - Start\ position.x)} \\ Y2 &= Start\ position.y + Index * (End\ position.y - Start\ position.y) \end{aligned}\right\} \quad (2)$$

$$\left.\begin{aligned} Index &= \frac{(obstacle\ location.y - Start\ position.y)}{(End\ position.y - Start\ position.y)} \\ X1 &= Start\ position.x + Index * (End\ position.x - Start\ position.x) \end{aligned}\right\} \quad (3)$$

$$\left.\begin{aligned} Index &= \frac{(obstacle\ location.y + obstacle.Height) - Start\ position.y}{(End\ position.y - Start\ position.y)} \\ X2 &= Start\ position.x + Index * (End\ position.x - Start\ position.x) \end{aligned}\right\} \quad (4)$$

Because X1, X2, Y1, and Y2 as values, are omitted in all functions, the task is examined in contrast to those that result across the examination of all obstacles. The planned route is used to achieve the target. If an intersection passes the obstacle, or if an intersection is not present, it will and if there is not, it will carry on moving towards the target.

Direct Or Reverse checks any direction go agent if have best time and shortest path.

The walk is executed by Timer1, and each step takes 0.01 seconds. It calculates the subsequent point ne until arriving at the final point; GetNextPoint is used to calculate the subsequent point. It then moves towards the final point using MoveAgent. On arrival at the first target, a sub-task is performed by the agent which is to seek the required object and match it with another on the bookcase. If it finds the object, then it will go to the table. Otherwise, it will continue to carry out the subsequent queued task.

In Table 3, a comparison of the three methods is exhibited: the shortest possible path, the most efficient time, and the negative methods.

Table 3. *Comparison between three techniques.*

Performance	When filling randomly	When using the Intelligent Code	When using a Heuristic Search
Competency	48%	97%	95%
Negatives	It takes a very long time.	For every object, there has to be an intelligent code. This must not change its position within the environment within which the object is constructed.	None

The most efficient method is the heuristic search. This is due to the fact it is unrelated to constraints. Changes to the environment are able to interplay with alterations and provide a faster result.

5. Conclusion

Environmental characteristics are crucial, particularly within the intelligent agents' program design. The initial step should identify the job's environment along with the maximum number of characteristics. Task environments are varied across many important dimensions. They are partly or completely observable, stochastic or deterministic, sequential or episodic, dynamic or static, continuous or discrete, and can be multi- or single-agent. Resultantly, it is recommended that construction starts with identical rules, regardless of the amount of time it takes. Progressive research methods are appropriate for intelligent agents facing life's function problems. This work presented the hierarchical structure of the environment for the proposed model library and presented the object that contains the library. The main algorithms that used for the proposed model is a knowledge-based system for one process in the queue, which mimics human behavior (a librarian in presented case). In addition, it used a heuristic search technique to find a quick solution which our results are achieved because it is not related to any constraints and any changes in the environment.

With respect to work in the future, considering complex

environments with stochastic and multi-agents is highly recommended. The environment should be collaborative, cooperative, and competitive and intelligent agents will work together as a cell (single unit).

References

[1] Zeigler, B. P. (2014). Object-oriented simulation with hierarchical, modular models: intelligent agents and endomorphic systems. Academic press.

[2] Wooldridge, M., & Dunne, P. E. (2001, August). The computational complexity of agent verification. In International Workshop on Agent Theories, Architectures, and Languages (pp. 115-127). Springer, Berlin, Heidelberg.

[3] Balduccini, M., & Lanzarone, G. A. (1997). Autonomous semi-reactive agent design based on incremental inductive learning in logic programming. Proc. of the ESSLI, 97, 1-12.

[4] Niels, B. A. (2002). *A model for Procedural Knowledge*, PhD thesis, University of Nyenrode, Breukelen.

[5] Thorndyke, P. W., & Goldin, S. E. (1983). Spatial learning and reasoning skill. In Spatial orientation (pp. 195-217). Springer, Boston, MA.

[6] Jennifer Herron (2017) Intelligent Agents for the Library, Journal of Electronic Resources in Medical Libraries, 14: 3-4, 139-144, DOI: 10.1080/15424065.2017.1367633.

[7] Den Heijer, F. M., & Goede, R. (2014). Implementing an intelligent agent in a known, observable, discrete and deterministic environment using a scriptable game-engine. In Intelligent Systems and Agents 2014 Conference (ISA 2014), in press, Lisbon, Portugal.

[8] Owaied, H. H., & Abu-A'ra, M. M. (2007, June). Functional Model of Human System as Knowledge Based System. In IKE (pp. 158-164).

[9] Barjtya, S., Sharma, A., & Rani, U. (2017). A detailed study of Software Development Life Cycle (SDLC) models. International Journal Of Engineering And Computer Science, 6 (7), 22097-22100.

[10] Abuhadba, S. (2011). An Intelligent Agent Model and a Simulation for a. Given Task in a Specific Environment. Supervisor. Dr. Hussein H. Owaied.

[11] Shah, S., Lynch, L. M., & Macias-Moriarity, L. Z. (2010). Crossword puzzles as a tool to enhance learning about anti-ulcer agents. American journal of pharmaceutical education, 74 (7).

[12] Chen, B., & Cheng, H. H. (2010). A review of the applications of agent technology in traffic and transportation systems. IEEE Transactions on intelligent transportation systems, 11 (2), 485-497.

[13] Bekey, G. A. (2005). Autonomous robots: from biological inspiration to implementation and control. MIT press.

[14] Dorf, Richard C., "Modern Control Systems, 7th edition" (1995). Books by Marquette University Faculty. Book 184. http://epublications.marquette.edu/marq_fac-book/184.

[15] McCorduck, P. (1983). The fifth generation: artificial intelligence and Japan's computer challenge to the world. Reading, Mass.: Addison-Wesley.

[16] Kumar, P. R., & Varaiya, P. (1986). Stochastic systems: estimation, identification and adaptive control (Prentice-Hall Information & System Sciences Series).