An Intelligent Architecture for Autonomous Virtual Agents Inspired by Onboard Autonomy

Kaveh Hassani and Won-Sook Lee

School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, Canada
{kaveh.hassani,wslee}@uottawa.ca

Abstract. Intelligent virtual agents function in dynamic, uncertain, and uncontrolled environments, and animating them is a chaotic and error-prone task which demands high-level behavioral controllers to be able to adapt to failures at lower levels of the system. On the other hand, the conditions in which space robotic systems such as spacecraft and rovers operate, inspire by necessity, the development of robust and adaptive control software. In this paper, we propose a generic architecture for developing autonomous virtual agents that let them to illustrate robust deliberative and reactive behaviors, concurrently. This architecture is inspired by onboard autonomous frameworks utilized in interplanetary missions. The proposed architecture is implemented within a discrete-event simulated world to evaluate its deliberative and reactive behaviors. Evaluation results suggest that the architecture supports both behaviors, consistently.

Keywords: Virtual agents. Autonomous systems. Cognitive architectures.

1 Introduction

Embodied agents have been extensively investigated in both robotics and virtual environments. In latter field, they are referred as intelligent virtual agents (IVA). Believability is a critical aspect of an IVA which can be augmented by surface realization and intelligent behavior. Although advances in computer graphics has led to realistic surface realization, yet the IVA's behavior is monotonous due to its scripted nature. Intelligent behavior emerges from cognitive characteristics such as recognition, decision making, perception, situation assessment, prediction, problem solving, planning, reasoning, belief maintenance, execution, interaction and communication, reflection, and learning [1].

Traditional IVA architecture follows a dualist perspective which decomposes the agent to a mind and a body. Mind as an abstract layer provides the agent with cognitive functionalities. It receives perceptions from the body, makes decisions, and sends the decisions in terms of abstract actions to the body. The body as an embodied layer animates the received actions within the virtual environment and provides the mind with perceptions acquired from its virtual sensors. Continues interaction between mind and body forms a closed perception-cognition-action loop. However, some recent studies challenge the strict separation between mind and body [2].

In terms of AI, Russell and Norving [3] utilized the notation of rational agents, and categorized them to simple reflex, model-based reflex, goal-based, and utility-based agents. In robotics literature, agents are classified to cognitive, behavioral and hybrid agents. Wooldridge [4] categorized intelligent agents to logic-based, reactive, belief-desire-intention (BDI) and layered agents. Among them, BDI agent has been widely utilized as an IVA architecture. Logic-based agents exploit symbolic logic deductions and cannot handle uncertainties. Cognitive agents such as BDI provide deliberative decision making capabilities for long temporal horizons. However, they cannot react to the situations which need instant responds. Furthermore, knowledge representation is a main challenge in these architectures. Reactive agents (i.e. behavior-based agents in robotics literature) couple the control and decision making mechanisms to the current local sensory information to provide real-time reactions. Although this approach minimizes the complexity of the representational structures and provides quick responses to dynamic environments, it is not scalable and suffers from the lack of reasoning capabilities and task-oriented behaviors. Hybrid agents have a layered structure. Layers function in different abstraction and operational frequency levels, and thus let the agent to combine reactive and deliberative behaviors.

An IVA functions in a dynamic, uncertain, and uncontrolled environment, and animating it is a chaotic and error-prone task which demands high-level behavioral controllers to be able to adapt to failure at lower levels of the system (e.g. when a navigation system fails to direct a walking agent to a desired waypoint). The conditions in which space robotic systems such as satellites, spacecraft and rovers operate, inspire by necessity, the development of robust and adaptive control software. These autonomous systems which have been successfully employed by NASA and ESA[1] can achieve mission goals and handle unpredicted situations, autonomously [5].

According to Muscettola et al. [6], challenges of developing agent architectures for onboard autonomy in space missions are driven by four major characteristics of the spacecraft as follows. First, the spacecraft must perform autonomous operations for long periods of time without human guidance. Second, the performed operations must guarantee success, given tight deadlines and resource constraints. Third, due to high cost of the spacecraft, its operations require high reliability. Fourth, spacecraft operation involves concurrent activities among a set of tightly coupled subsystems.

In this paper, we adopt Remote Agent (RA) architecture [6] (i.e. developed by NASA to autonomously control the DS-1 spacecraft as part of New Millennium Deep Space Mission-1 to flyby an asteroid) to develop a generic architecture for IVA development. Furthermore, we utilize a fuzzy ontology as the agent's knowledge representation scheme. Adopting RA architecture reinforces IVA by a reliable and intelligent platform that has already shown to be successful in complex inter-planetary missions. Moreover, embedded fuzzy ontology lets IVA to acquire knowledge from environmental uncertainties and construct a proper belief model. This paper is organized as follows: in section 2 an overview of related works is presented. In section 3, we describe our proposed architecture. In section 4, experimental results and evaluations are discussed. Finally, section 5 concludes the paper.

[1] European space agency

2 Related Works

According to Langley, Laird and Rogers [1] "A cognitive architecture specifies the underlying infrastructure for an intelligent system". During the last decades, several cognitive architectures have been proposed. ACT-R [7] cognitive framework emphasizes human psychological verisimilitude. Soar [8] as a rule-based cognitive architecture formulates the tasks as attempts to achieve goals. ICARUS [9] model designed for embodied agents emphasizes perception and action over abstract problem solving. SASE [10] is based on Markov decision processes, and utilizes the concept of autonomous mental development. PRODIGY [11], DUAL [12] and Polyscheme [13] are other examples of cognitive architectures. Probably, BDI [14] is the most representative model of cognitive agents. It triggers behaviors driven by conceptual models of intentions and goals in complex dynamic scenarios. BBSOAA [15] is an extension of BDI architecture that enhances the knowledge representation and inference capabilities, and is suitable for simulating virtual humans. Although BDI-inspired architectures such as IRMA [16] support long term behaviors, their current implementations are whether hardware-based or logic-based. More information regarding cognitive architectures can be found in [1].

A few IVA architectures concern with software engineering issues. As an instance, CAA [17] is a generic object-oriented architecture that supports context-sensitive behaviors. Additionally, A few architectures such as CMION [18] are developed based on the notion of migrating agent which refers to the ability of an agent to morph from one form of embodiment to another. A few studies such as embodied cognition model [19] challenge the strict separation between mind and body. This model embeds a secondary control loop, subconscious mind, into the body layer. Moreover, some research works emphasize on machine learning techniques to enhance the robustness. Reinforcement learning for behavioral animation [20] and FALCON-X [21], an IVA learning architecture that utilizes self-organizing neural model, are examples of these studies. OML [22] is an agent architecture for virtual environments equipped with neural network-based learning mechanism. In this model, a sensory neuron represents an object, and a motor neuron represents an action. An alternative paradigm for developing IVA is to employ a middleware to integrate existing multi-agent systems such as 2APL [23], GOAL [24], Jadex [25] and Jason [26] with existing game engines. This systematic approach benefits from reusability and rapid prototyping characteristics. As an example, CIGA [27] is a middleware that amalgamates an arbitrary multi-agent system with a game engine by employing domain ontology.

A few IVA architectures, similar to behavioral robotic frameworks, investigate the behavioral organization and action selection. SAIBA [28] is a popular framework that defines a pipeline for abstract behavior generation. It consists of intent planner, behavior planner and behavior realizer. Thalamus [29] framework adds a perceptual loop to SAIBA to let the embodied agent to perform continuous interaction. AATP [30] is a coupled planning and execution architecture for action selection in cognitive IVAs. Neural-dynamic architecture [31] utilizes a dynamic neural field to describe and learn the behavioral state of the system, which in turn, enables the agent to select the appropriate action sequence regarding its environment.

Ultimately, layered architectures (i.e. hybrid models) perform deliberative and reactive operations, simultaneously. COGNITIVA [32] is a reactive-deliberative agent architecture that consists of reactive, deliberative and social layers. In those situations that there is no time for planning, the reactive layer reacts to the situation. Otherwise, the architecture generates goals, plans sequence of actions to reach those goals, schedules the actions, and executes them. Hybrid architectures are widely utilized in space robotic systems as well [5]. RA [6] is a hybrid architecture tested on deep space-1. It is designed to provide reliable autonomy for extended periods. IDEA [33] is a multi-agent architecture that supports distributed autonomy by separating the layers of architecture to independent agents. IDEA has been successfully evaluated on K9 rover. MDS [34] is a hybrid software framework that emphasizes state estimation and control whereas TITAN [34] emphasizes model-based programming. LAAS [35] and Claraty [36] are other examples of hybrid architectures utilized in space missions. The modern space systems including satellite systems (e.g. EO-1 and Techsat-21) and interplanetary missions (e.g. DS-1) exploit hybrid architectures. We adopt RA framework as a reliable architecture to design a generic architecture for autonomous virtual agents that consistently supports deliberative and reactive behaviors.

3 Autonomy for Virtual Agents

RA [6] architecture provides the spacecraft with onboard autonomy and is developed as a hybrid platform with three operational layers including: deliberative planner-scheduler (PS), reactive executive, and mode identification and recovery system (MIR). PS determines the optimal execution sequence of actions in a way that spacecraft can reach its predefined mission goals. Also, it schedules the start time of the actions. Reactive executive receives scheduled actions from PS, and decomposes them to sub-actions understandable by flight software. Flight software is an interface between RA and spacecraft hardware, and consists of collection of software packages such as motor controllers managed by RA. Moreover, executive monitors the execution process to detect the inconsistencies in plans. MIR consists of mode identification (MI) and mode recovery (MR) units. MI transfers the low-level sensor data to high-level perceptions and provides its upper levels with the current system configuration. Ultimately, MR provides the system with error detection and recovery services.

Schematic of our proposed architecture, inspired by RA, is shown in Figure 1. It consists of two layers including cognitive and executive layers. Furthermore, it utilizes a middleware as an interface between abstract agent and its embodied counterpart animated by a game engine. In this architecture, the components are placed in their corresponding layers regarding their operational frequency and abstraction level. The cognitive layer is responsible for providing cognitive functionalities whereas the executive layer is responsible for executing the decisions made by cognitive layer and providing the cognitive layer with high level feedbacks. Cognitive layer functions in low frequency and high level knowledge representation, and plans for long temporal horizons whereas executive layer functions in high frequency and deals with the current situations in a reactive and soft real-time manner.

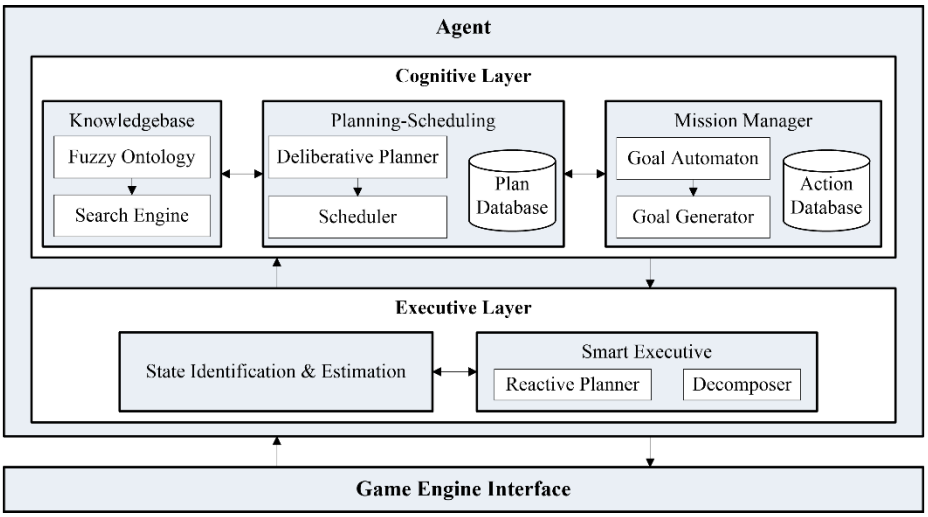

Fig. 1. Schematic of the proposed generic architecture for autonomous virtual agents

3.1 Cognitive Layer

Cognitive layer provides IVA with autonomy, and consists of three components including mission manager (MM), planning-scheduling (PS) and knowledgebase (KB). MM contains the agent's goals and feasible actions. It consists of three sub-units including goal automaton, goal generator, and action database. Goal automaton keeps a network of predefined goals, and is defined as a DFA (deterministic finite automaton) $A=<Q,\Sigma,\sigma,q,F>$ where Q denotes a set of goals, Σ is the evaluation signal indicating whether the current goal has been achieved, σ is the transition function (i.e. $\sigma:Q\times\Sigma\rightarrow Q$) which determines the priority of goals, $q\epsilon Q$ determines the initial goal, and $F\subseteq Q$ is the set of final goals. Structurally, goal automaton is a graph whose nodes present the goals, and edges determine the satisfaction criteria of corresponding goals.

Goal generator functions as the transition function of the goal network. In each time step, it evaluates current goal and received perceptions in order to determine whether the current goal is satisfied. If so, it transforms the goal state to a new goal within the goal automaton, and sends the new goal to PS, so that it can plan new sequence of actions. In case that the goal generator detects the current goal is not satisfied, it keeps the current goal as mission objective.

Regarding the physical constraints of controlled system, there is a limited set of valid actions that agent can execute. These feasible actions are stored in action database. An action is a high-level abstract activity that encapsulates a few low-level sub-actions and consists of a unique identifier, some preconditions and effects, estimated execution duration, set of sub-actions, and their execution timeline. This abstraction scheme dramatically reduces the complexity of planning and scheduling processes. Preconditions determine the constraints on state variables which must be satisfied in order to an action

can be executed. Effects determine how the execution of an action affects the state variables. Planner relies on information regarding preconditions and effects of actions to determine the optimal sequence of actions.

PS plays a crucial role in the proposed architecture. It consists of three sub-units including deliberative planner, scheduler and plan database. Deliberative planner decides serial or parallel sequences of actions fetched from action database for long temporal horizons to reach the mission objectives in an optimal trajectory based on the perceptions received from executive layer, goals fetched from mission manager, and required information by actions from knowledgebase. It utilizes a backtracking algorithm with pruning strategy to find the best sequence of actions that achieve the current goal. The backtracking algorithm constructs a valid and optimal sequence based on information regarding the effects and preconditions of the actions. It is noteworthy that pruning strategy reduces both spatial and temporal complexities, significantly.

As soon as deliberative planner completes the planning process, it sends the action sequence to scheduler, which in turn, determines the start time of the sequence. Estimated execution time of each action is computed using regression techniques. Using this information, scheduler assigns a start time to each action within the sequence. Then, the planned and scheduled action sequence is inserted into the plan database. In each time step, this temporal database retrieves actions regarding their start time and sends them to the executive layer, which in turn, executes them.

KB component as a profound memory provides the agent with knowledge acquired from perception sequence. Essentially, a knowledgebase consists of a set of sentences that claim something about the world, an updating mechanism, and a knowledge extraction engine [1]. Our KB consists of two sub-modules: fuzzy ontology and search engine. Fuzzy ontology represents the concepts, objects, features and their relations based on the agent's perceptional history. The ontology can be constructed either in design-time to keep the built-in knowledge, or in run-time to automatically capture the knowledge, or in a hybrid manner. It utilizes a maintainer as an updating mechanism that receives current perceptions from the executive layer and compares them with the knowledge represented in the ontology. Based on this comparison, it may decide to insert new concepts, objects or relations, update them, or even prune the ontology to omit the redundancies or inconsistencies. It is noteworthy that extending the ontology with fuzzy theory enables the agent to model both internal and external uncertainties. We utilize the fuzzy ontology proposed in [37] to design the agent's knowledgebase. Search engine receives queries from PS and searches the ontology by applying iterative first depth search. Then, it returns the resultant knowledge to PS.

3.2 Executive Layer

Executive layer executes the decisions made by cognitive layer, monitors the execution process, and provides the cognitive layer with high-level feedbacks. As illustrated in Figure 1, this layer consists of two main components including state identification and estimation unit, and smart executive. The first component, state identification and estimation unit is responsible for providing the framework with perceptions and estimations. It receives the sensory data from the game engine interface and maps it to the

formal knowledge representation used by cognitive and executive layers. In other words, it converts data acquired from IVA's virtual sensors to the perceptions cognoscible by the agent. In order to complete this task, it utilizes Kalman filters for data assimilation and fuzzifiers for data conceptualization. Moreover, it can exploit variety of software libraries to perform specialized data processing activities such as automatic speech recognition, image processing, etc. Therefore, state identification and estimation unit enables the agent to deal with a variety of sensory data acquired from different sensory channels.

Smart executive is responsible for executing sequences of planned actions within plan database, and monitoring the execution process in order to prevent inconsistencies. It consists of two sub-components including decomposer and reactive planner. Decomposer fetches the scheduled action sequences from the plan database, assigns a software thread to each of the retrieved actions, and starts the threads according to the schedule. Using this approach, agent can perform parallel plan execution. As aforementioned, each action is an abstract activity that embodies a set of low-level activities (i.e. sub-actions). In the beginning of execution of an action, it invokes its corresponding sub-actions according to a predefined timeline. This timeline is a built-in knowledge defined by system experts. Execution of each sub-action results in an activity in embodied layer (i.e. agent's avatar). Thus, using this hierarchal scheme, abstract decisions are mapped to physical manipulations and actuations within the virtual environment. Furthermore, decomposer can employ specialized software libraries to provide the actions with required facilities such as text-to-speech engine. Additionally, decomposer monitors the execution to prevent inconsistencies. It compares the current states of the system with the expected states predicted by state estimation, and in case of any irregularities, it halts the inconsistent thread and sends a signal to the reactive planner so that it can take a proper action to eliminate the inconsistency.

Reactive planner is an event-oriented planner that reacts to the unpredicted situations. It exploits a greedy algorithm to choose an action that can handle the unpredicted situation with minimum cost. The cost is computed based on the number of actions required to be performed in order to return the agent's configuration to the planned trajectory. Therefore, in case of unpredicted situation, reactive planner switches from monitoring mode to reacting mode and executes minimal number of activities to solve the inconsistency. It is noteworthy that in those situations that reactive planner executes an unplanned action, it informs PS regarding the divergence in plan sequence, which in turn, investigates whether the current sequence can still reach the current goal. If the goal is not reachable, it repairs the sequence using a backtracking algorithm, and informs the decomposer to fetch the repaired sequence to execute.

Ultimately, game engine interface as a middleware provides an interface between the abstract agent (i.e. mind) and its embodied counterpart (i.e. body). It directly maps the sub-actions to physical activities within the virtual environment. Moreover, it transfers the virtual sensory data from the avatar to its controlling layer. Additionally, it performs a few pre-processing steps such as encoding and decoding data to facilitate the coordination between the abstract and animated layers. It is noteworthy that our proposed architecture is body-independent in a sense that it can adapt to a given body by embedding proper knowledge within its mission manager.

4 Experimental Results

Reliability of our proposed architecture is partially supported by evaluation results of its predecessors operating in inter-planetary missions. However, for further evaluations, we design a discrete event-based 2D world-- the world of circles. In this virtual world, we define three different types of circles regarding their colors (i.e. red, green and blue). These circles may either pop into the world randomly or in a predefined order. The only constraint is that they can only enter the world from a fixed position called the origin. In case of random entrance, a circle with random color enters the world from the origin point. Otherwise, the order of circles is announced before their arrival. There are three non-stationary target zones in the world, each corresponding to a particular color. Also, a few non-stationary obstacles are embedded in the world. The agent's goal in this simulation is to pick up a circle from the origin and move it to one of the target zones regarding the color similarity while avoiding the obstacles (e.g. blue circles to blue target zone). It is noteworthy that because the random circles can pop into the world among the predefined sequence of circles, and both targets and obstacles are non-stationary, the world provides proper characteristics to evaluate both reactive and deliberative behaviors.

We applied our proposed architecture to the world of circles by implementing an agent based on our architecture to move the circles to their target zones while avoiding the obstacles. The agent and environment are implemented in C#.Net programming language. A sample scene of the simulation is shown in Figure 2. The gray rectangles present obstacles whereas blue, purple and green rectangles present the target zones. In this simulation, user can determine the number of circles, their colors, and the randomness level (i.e. randomness of 0.1 means one out of every ten circles pops into the world randomly following a uniform distribution). The agent's KB contains the positions of the origin, obstacles and target zones as variables. Its action database contains the following action:

```
Action identifier: Move (Circle)
Preconditions:
           Status: Idle
          #Circles>0
Effects:
           #Circles--
Estimated execution time: #Steps
Sub-Actions:
           Status: Moving
           Fetch(Circle);
           Loop: FindPath(Circle);
           Move(Circle);
           If(Failure)
               FindTarget(Circle); Update(KB);Goto Loop;
           Else
               Status: Idle
```

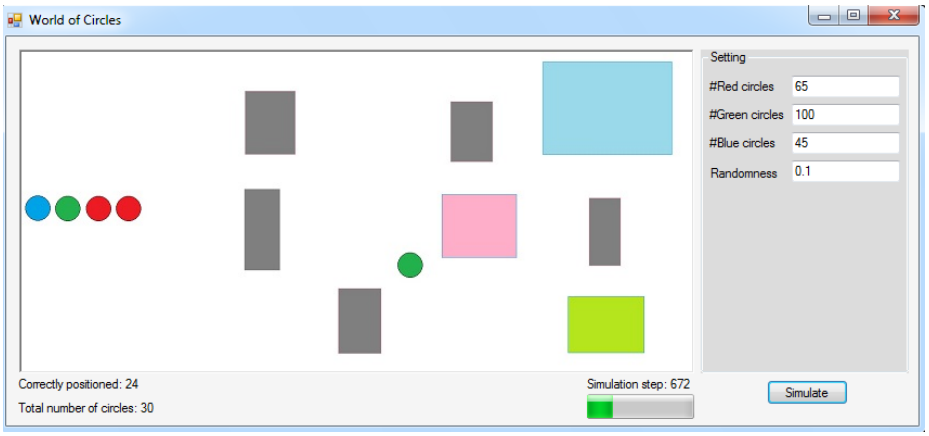

Fig. 2. Snapshot of discrete event implementation of world of circles

Preconditions of this action state that the agent must be idle and there must be new circles in queue. This action first moves the agent to the origin position and then finds an optimal path from origin point to the corresponding target zone using A^* algorithm based on current values of position variables. Then, it moves the front circle in queue to its target zone. If it succeeds, the action is completed. Otherwise, it finds the new position of the target, updates its knowledgebase and repeats the path planning process. As soon as the agent gets information regarding the arrival of new sequence, its deliberative planner plans paths for those circles whose arrival is announced. However, it is possible that a random circle arrives among the predetermined circles. In this case, the reactive planner activates, plans the proper path for the random circle, and then informs the PS. Then, the scheduler reschedules the remaining actions.

In order to facilitate the evaluations, we employ timing diagrams of simulation process to represent the activation sequences of components. A sample timing diagram of the simulations is shown in Figure 3 in which vertical axis indicates the components (i.e. FO: fuzzy ontology, SE: search engine, PS: planner-scheduler, RP: reactive planner, MM: mission manager, SIE: state identification and estimation, DE: decomposer). As shown in Figure 3, components function in different frequencies. As an example, while agent is executing the plans from $T_{PS:1}$, a random circles pops in and reactive planner reacts to the situation in $T_{RP:1}$.

We run the simulations for 1100 times. After each 100 simulations, we increase the randomness by 0.1. Also, we use 200 circles with uniform random colors in each trial. The length of a sequences is selected by a random uniform distribution in range of [5,25]. Simulation results are shown in Figure 4. As illustrated, by increasing the randomness of the simulation, the number of activations of deliberative planner decreases whereas this number increases for reactive planner. Furthermore, even when there is no randomness in the simulation, the reactive planner activates when two consecutive sequences appear with a small delay. In this case, deliberative planner does not have enough time to plan for all the circles, thus agent detects them as unplanned circles, and activates the reactive planner.

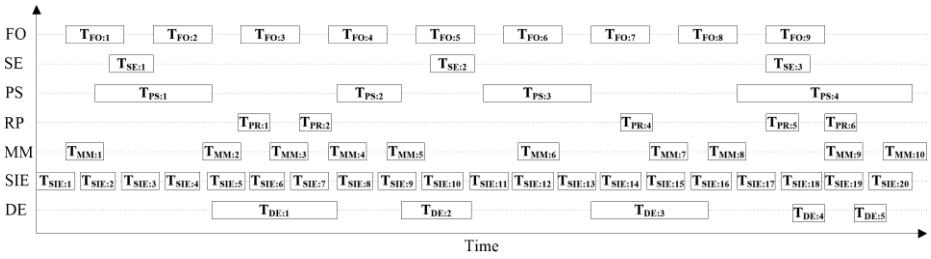

Fig. 3. Sample timing diagram of activation of components within the proposed architecture

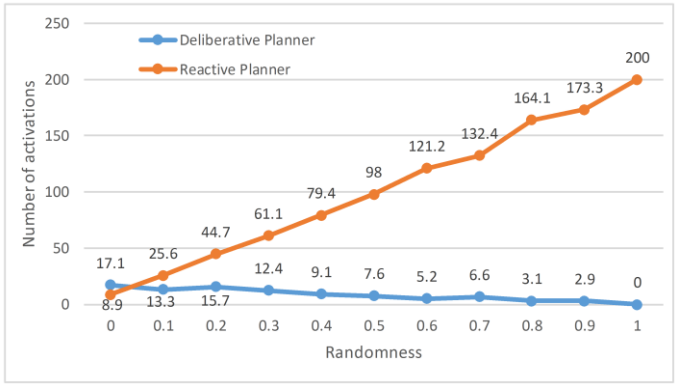

Fig. 4. Average number of reactive and deliberative behaviors in response to changes in randomness of the simulations

5 Conclusion

In this paper, we introduced a generic architecture for autonomous virtual agents inspired by onboard autonomy utilized in inter-planetary space missions. Our proposed architecture provides the agent with concurrent deliberative and reactive behaviors. Furthermore, it equips the agent with necessary components for autonomy such as fuzzy knowledgebase and smart executive. In order to validate our proposed architecture, we implemented a discrete event simulated world. The evaluation results of applying our agent architecture on this world suggest that the architecture is valid and consistent, and is able to handle deliberative and reactive functionalities, simultaneously. Moreover, it can properly support the required parallelism among the processes. As future works, we are planning to apply our agent architecture to a complex game scenario and investigate it is capability in playing the game autonomously. Moreover, we are planning to exploit reinforcement learning so that agent can learn its feasible actions, independently. Ultimately, we are planning to utilize a self-organizing neuro-fuzzy architecture to learn the ontology automatically from perception sequence. Using these two learning schemes renders the need for expert's knowledge obsolete.

References

1. Langley, P., Laird, J.E., Rogers, S.: Cognitive Architectures: Research Issues and Challenges. Cogn. Syst. Res. 10, 141–160 (2009)
2. Ribeiro, T., Vala, M., Paiva, A.: Censys: A Model for Distributed Embodied Cognition. In: 13th International Conference on Intelligent Virtual Agents, pp. 58–67. Springer (2013)
3. Russell, S., Norving, P.: Artificial Intelligence: A Modern Approach. Prentice Hall, New Jersey (2010)
4. Wooldridge, M.: Intelligent Agents: The Key Concepts. In: Marik, V., Stepankova, O., Krautwurmova, H., Luck, M. (eds.) Multi-Agent Systems and Applications II, pp. 3–43. Springer (2002)
5. Hassani, K., Lee., W-S.: A Software-in-the-Loop Simulation of an Intelligent Micro-Satellite within a Virtual Environment. In: IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications, pp. 31–36. IEEE Press (2013)
6. Muscettola, N., Nayak, P., Pell, B., Williams., B.: Remote Agent: To Boldly Go Where no AI System has Gone Before. Artificial Intell. 103, 5–48 (1998)
7. Anderson, J., Bothell, D., Byrne, M., Douglass, S., Lebiere, C., Qin, Y.: An Integrated Theory of the Mind. Psych. Rev. 111, 1036–1060 (2004)
8. Laird, J., Newell, A., Rosenbloom, P.: Soar: An Architecture for General Intelligence. Artificial Intell. 33, 1–64 (1987)
9. Langley, P., Choi., D.: A Unified Cognitive Architecture for Physical Agents. In: 21st National Conference on Artificial Intelligence, pp. 1469–1474. AAAI Press (2006)
10. Weng, J.: Developmental Robotics: Theory and Experiments. Int. J. Humanoid Robot. 1, 199–236 (2004)
11. Carbonell, J., Etzioni, O., Gil, Y., Joseph, R., Knoblock, C., Minton, S., Veloso, M.: PRODIGY: An Integrated Architecture for Planning and Learning. In: Lehn, K. (eds.) Architectures for Intelligence, pp. 51–55. ACM Press (1991)
12. Kokinov, B: The DUAL Cognitive Architecture: A Hybrid Multi-Agent Approach. In: 11th European Conference of Artificial Intelligence, pp. 203–207. ECAI (1994)
13. Cassimatis, N., Nicholas, L.: Polyscheme: A Cognitive Architecture for Integrating Multiple Representation and Inference Schemes. MIT Ph.D. Dissertation (2002)
14. Rao, A., Georgeff., M.: BDI-agents: From Theory to Practice. In: 1st International Conference on Multi-agent Systems, pp. 312–319. ICMAS (1995)
15. Liu, J., Lu., Y.: Agent Architecture Suitable for Simulation of Virtual Human Intelligence. In: 6th World Congress on Intelligent Control and Automation, pp. 2521–2525. IEEE Press (2006)
16. Bratman, M., Israel, D., Pollack, M.: Plans and Resource-Bounded Practical Reasoning. Comput. Intell. 4, 349–355 (1988)
17. Kim, I.: CAA: A Context-Sensitive Agent Architecture for Dynamic Virtual Environments. In: 5th International Conference on Intelligent Virtual Agents, pp. 146–151. Springer (2005)
18. Kriegel, M., Aylett, R., Cuba, P., Vala, M., Paiva, A.: Robots Meet IVAs: A Mind-Body Interface for Migrating Artificial Intelligent Agents. In: 10th International Conference on Intelligent Virtual Agents, pp. 282–295. Springer (2011)
19. Vala, M., Ribeiro, T., Paiva, A.: A Model for Embodied Cognition in Autonomous Agents. In: 12th International Conference on Intelligent Virtual Agents, pp. 505–507. Springer (2012)

20. Conde, T., Tambellini, W., Thalmann, D.: Behavioral Animation of Autonomous Virtual Agents Helped by Reinforcement Learning. In: 4th International Workshop on Intelligent Virtual Agents, pp. 175–180. Springer (2003)
21. Kang, Y., Tan, A.: Self-Organizing Cognitive Models for Virtual Agents. In: 13th International Conference on Intelligent Virtual Agents, pp. 29–43. Springer (2013)
22. Wibner, M.: Simulation of a Motivated Learning Agent. In: International Workshop on Agents for Educational Games and Simulations, pp. 151–165. Springer (2012)
23. Dastani, M.: 2APL: A Practical Agent Programming Language. Auton. Agents and Multi-Agent Sys. 16: 214–248 (2008)
24. Hindriks, K.: Programming Rational Agents in GOAL. In: Seghrouchni, A., Dix, J., Dastani, M., Bordini, R. (eds.) Multi-Agent Programming: Languages, Tools and Applications, pp. 119–157. Springer (2009)
25. Pokahr, A., Braubach, L., Lamersdorf, W.: Jadex: A BDI Reasoning Engine. In: Bordini, R., Dastani, M., Dix, J., Seghrouchni, A. (eds.) Multi-Agent Programming: Languages, Platforms and Applications, pp. 149–174. Springer (2005)
26. Bordini, R., Hübner., J., Wooldridge, M.: Programming Multi-Agent Systems in AgentSpeak Using Jason. John Wiley (2007)
27. Oijen, J., Vanhee, L., Dignum, F.: CIGA: A Middleware for Intelligent Agents in Virtual Environments. In: International Workshop on Agents for Educational Games and Simulations, pp. 22–37. Springer (2012)
28. Kopp, S.: Towards a Common Framework for Multimodal Generation: The Behavior Markup Language. In: 6th International Conference on Intelligent Virtual Agents, pp. 205–217. Springer (2006)
29. Ribeiro, T., Vala, M., Paiva, A.: Thalamus Closing the Mind-Body Loop in Interactive Embodied Characters. In: 12th International Conference on Intelligent Virtual Agents, pp. 189–195. Springer (2012)
30. Edward, L., Lourdeaux, D., Barthes, J.: An Action Selection Architecture for Autonomous Virtual Agents. In: Nguyen, N., Katarzyniak, R., Janiak, A. (eds.) New Challenges in Computational Collective Intelligence, pp. 269–280. Springer (2009)
31. Sandamirskaya, Y., Richtert. M., Schoner, G.: A Neural-Dynamic Architecture for Behavioral Organization of an Embodied Agent. In: IEEE International Conference on Development and Learning, pp.1–7. IEEE Press (2011)
32. Spinola, J., Ricardo, I.: A Cognitive Social Agent Architecture for Cooperation in Social Simulations. In: 12th International Conference on Intelligent Virtual Agents, pp. 311–318. Springer (2012)
33. Muscettola, N., Dorais, G., Fry, C., Levinson, R., Plaunt, C.: IDEA: Planning at the Core of Autonomous Reactive Agents. In: 3rd International NASA Workshop on Planning and Scheduling for Space. NASA (2002)
34. Horvath, G., Ingham, M., Chung, S., Martin, O.: Practical Application of Model-Based Programming and State-Based Architecture to Space Missions. In: 2nd IEEE Conference on Space Mission Challenges for Information Technology, pp. 80–88. IEEE Press (2006)
35. Alami, R., Chautila, R., Fleury, S., Ghallab, M., Ingrand, F.: Architecture for Autonomy. Int. J. Robot. Res. 17, 315–337 (1998)
36. Nesnas, I., Simmons, R., Gaines, D., Kunz, C., Calderon, A., Estlin, T., Madison, R., Guineau, J., McHenry, M., Shu, I., Apfelbaum, D.: CLARAty: Challenges and Steps toward Reusable Robotic Software. Int. J. Advance Robot. Sys. 3, 23–30 (2006)
37. Hassani, K., Nahvi, A., Ahmadi. A.: Architectural Design and Implementation of Intelligent Embodied Conversational Agents Using Fuzzy Knowledgebase. J. Intell. Fuzzy Sys. 25, 811–823 (2013)