On Designing Migrating Agents: From Autonomous Virtual Agents to Intelligent Robotic Systems

Kaveh Hassani[*] and Won-Sook Lee[†]
School of Electrical Engineering and Computer Science, University of Ottawa

Abstract

In the realm of multi-agent systems, migration refers to the ability of an agent to transfer itself from one embodiment such as a graphical avatar into different embodiments such as a robotic android. Embodied agents usually function in a dynamic, uncertain, and uncontrolled environment, and exploiting them 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. 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 migrating and autonomous agents inspired by onboard autonomy which enables the developers to tailor the agent's embodiment by defining a set of feasible actions and perceptions associated with the new body. Evaluation results suggest that the architecture supports migration by performing consistent deliberative and reactive behaviors.

CR Categories: I.2.0 [Artificial Intelligence]: General–Cognitive simulation; I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence–Intelligent agents; I.2.4 [Artificial Intelligence]: Problem Solving, Control Methods, and Search–Plan execution, formation, and generation; Scheduling; I.2.9 [Artificial Intelligence]: Robotics– Autonomous vehicles;

Keywords: Migrating Agents, Intelligent Virtual Agents, Autonomous Systems, Cognitive Architectures.

1 Introduction

Embodied intelligence is a prominent topic in multi-agent systems and refers to a coupled mind-body loop in which high level deliberative processes working with symbolic representation of the world within the mind decide the behavior of the agent based on a collection of physical or virtual sensors and actuators within the body. Mostly, embodied intelligence 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 performs the received actions within the environment and provides the mind with perceptions acquired from its sensors. Continues interaction between mind and body forms a closed perception-cognition-action loop. However, embodied cognition strongly suggests that in humans, mind and body are coupled and cannot be separated. A few agent architectures challenge the strict separation between mind and body [Ribeiro et al. 2013]. As an example, embodied cognition model embeds a secondary control loop, subconscious mind, into the body layer [Vala et al. 2012]. Furthermore, in situated agents, embodiment is considered as a situational coupling between an agent and its environment [Steels and Brooks 1993].

Embodied agents have been extensively investigated in both physical world as robots and in virtual environments as intelligent virtual agents (IVA). These embodiments have been progressed later by embedding social context and human interactions to the social robotics [Fong et al. 2002] and embodied conversational agents (ECA) [Hassani et al. 2013 (b)], respectively. Regardless of uncanny valley effect [Mori 2012], believability is an important characteristic of an embodied agent in the social context and can be augmented by surface realization and intelligent behavior. Although advances in computer graphics have led to realistic surface realization in animated agents, yet physical constraints and limitation on robotic systems necessitates much more scientific effort. In terms of intelligent behavior, generally the embodied agents' 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 [Langley et al. 2009].

In robotics literature, embodied agents can be classified into cognitive, behavioral and hybrid agents [Siegwart et al. 2011]. Wooldridge [2002] categorized intelligent agents to logic-based, reactive, belief-desire-intention (BDI) and layered agents. 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 immediate responses. 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. Russell and Norving [2010] utilized the notation of rational agents and categorized them into simple reflex, model-based reflex, goal-based, and utility-based agents.

A new paradigm in embodied agents that is introduced recently is migrating agents. Migration refers to the ability of an abstract

[*] kaveh.hassani@uottawa.ca
[†] wslee@uottawa.ca

entity to morph from one embodiment into a different embodiment and control the new body, accordingly. A more specific definition of migration is introduced based on social companions regardless of the body they reside in. In this definition, the intrinsic problem of migration is how to preserve the identity of one companion inhabiting different embodiments from the user perspective [Arent and Kreczmer 2013], or basically find out what exactly migrates. In order to answer this fundamental question Kriegel et al. [2011] define the companion's identity as those features that persist and make it unique and recognizable from the user's perspective. These features include a set of goals, a set of emotional reaction rules, the companion's action tendencies, emotional thresholds and decay rates for each of the emotions types, the set of definitions of responses to orders from the mind and competencies styles of action, and functionalities parameters. Migrating companions requires by necessity sufficient level of abstraction, modularity, flexible definition of identity, and multiple platforms.

Embodied agents usually function in a dynamic, uncertain, and uncontrolled environment, and exploiting them 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 can achieve mission goals and handle unpredicted situations, autonomously [Hassani and Lee 2013]. 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 [Muscettola et al. 1998].

In this paper, we define migrating agents in more general sense. We define a migrating agent as an agent that is able to move its mind and mind-body interface among different embodiments and for that purpose only requires high level knowledge of actions and perceptions that new body is able to perform and percept, respectively. Thus, our goal is to design a generic autonomous architecture that can reside in different bodies without requiring customization. To do so, we propose a generic embodied agent architecture inspired by remote agent (RA) (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) [Muscettola et al. 1998] and intelligent distributed execution architecture (IDEA) [Muscettola et al. 2002] agent architectures. Furthermore, we utilize a fuzzy ontology as the agent's knowledge representation scheme. Adopting RA and IDEA architectures reinforces the architecture by a reliable and intelligent platform that has already proven to be successful in complex inter-planetary missions. Moreover, embedded fuzzy ontology lets the agent to acquire knowledge from environmental uncertainties and construct a proper belief model. Furthermore, the embodiment of the proposed architecture can be tailored by defining set of possible actions and perceptions associated with the new body.

The 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.

2 Related Work

Migrating agents have been investigated in literature in the context of social companions and human-robot interactions. In this context, agent's high level characteristics such as its emotions and behaviors, and its low level identities such as its voice transfer among different embodiments in a way that users can recognize the agent in different embodiments. The CMION architecture designed for the migrating companions of the LIREC project is an open source architecture for coordinating the sensors and effectors of such an agent with its mind. The architecture is designed to work with virtual graphical agents, including those on mobile devices, as well as robots. The ultimate goal of LIREC project is to move beyond the novelty effect of both social robots and ECAs towards social companions that can play an acceptable long term role. It is reported that user studies suggest the success of this architecture [Kriegel et al. 2011; Aylett et al. 2013]. Sarah is a LIREC-based companion that can be embodied in a robot, on a large graphical screen or in a handheld device. [Segura et al. 2012]. In [Gomes et al. 2011] an artificial pet with two robotic and graphical embodiments is proposed. In both embodiments behavior is driven by needs that are used to maintain coherence and motivate user interaction. In [Arent and Kreczmer 2013] indirect migration as a type of migration that lets the user to recognize the identity of companion in the new embodiment is introduced. It refers to the migration that goes through a third embodiment that has common components with the previous two. The user studies suggest that user can properly recognize the identity of this companion with a certain level of confidence, in some context.

In order to design a body-independent generic architecture, one should investigate cognitive architectures which specify the underlying infrastructure for an intelligent system [Langley et al. 2009]. Several well-known cognitive architectures have been introduced in the literature during the last decades. Soar [Laird et al. 1987] is a rule-based cognitive architecture that formulates the tasks as attempts to achieve goals. ACT-R [Anderson et al. 2004] cognitive framework emphasizes human psychological verisimilitude. ICARUS [Langley and Choi 2006] model designed for embodied agents emphasizes perception and action over abstract problem solving. SASE [Weng 2004] is based on Markov decision processes and utilizes the concept of autonomous mental development. PRODIGY [Carbonell et al. 1991], DUAL [Kokinov 1994] and Polyscheme [Cassimatis and Nicholas 2002] are other examples of cognitive architectures. Probably, BDI [Rao and Georgeff 1995] is the most representative model of cognitive agents. It triggers behaviors driven by conceptual models of intentions and goals in complex dynamic scenarios. BBSOAA [Liu and Lu 2006] 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 [Bratman et al. 1998] support long term behaviors, their current implementations are whether hardware-based or logic-based. More information regarding cognitive architectures can be found in [Langley et al. 2009].

A few agent architectures concern with software engineering issues. As an instance, CAA [Kim 2005] is a generic object-oriented architecture that supports context-sensitive behaviors. Moreover, some research works emphasize on machine learning techniques to enhance the robustness. Reinforcement learning for behavioral animation [Conde et al. 2003] and FALCON-X [Kang and Tan 2013], an IVA learning architecture that utilizes self-organizing neural model are examples of these studies. OML [Wibner 2012] 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 graphical agents is to employ a middleware to integrate existing multi-agent systems such as 2APL [Dastani 2008], GOAL [Hindriks 2009], Jadex [Pokahr et al. 2005] and Jason [Bordini et al. 2007] with existing game engines. This systematic approach benefits from reusability and rapid prototyping characteristics. As an example, CIGA [Oijen et al. 2012] 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 [Kopp 2006] is a popular framework that defines a pipeline for abstract behavior generation. It consists of intent planner, behavior planner and behavior realizer. Thalamus [Ribeiro et al. 2012] framework adds a perceptual loop to SAIBA to let the embodied agent to perform continuous interaction. AATP [Edward et al. 2009] is a coupled planning and execution architecture for action selection in cognitive IVAs. Neural-dynamic architecture [Sandamirskaya et al. 2011] 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 [Spinola and Ricardo 2012] 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 [Hassani and Lee 2013 (a)]. RA [Muscettola et al. 1998] is a hybrid architecture tested on deep space-1. It is designed to provide reliable autonomy for extended periods. IDEA [Muscettola et al. 2002] 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 [Horvath et al. 2006] is a hybrid software framework that emphasizes state estimation and control whereas TITAN [Horvath et al. 2006] emphasizes model-based programming. LAAS [Alami et al. 1998] and Claraty [Nesnas et al. 2006] 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 and IDEA frameworks to design a generic architecture for autonomous virtual agents with migrating capabilities that consistently supports deliberative and reactive behaviors.

RA [Muscettola et al. 1998], shown in Figure 1, 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).

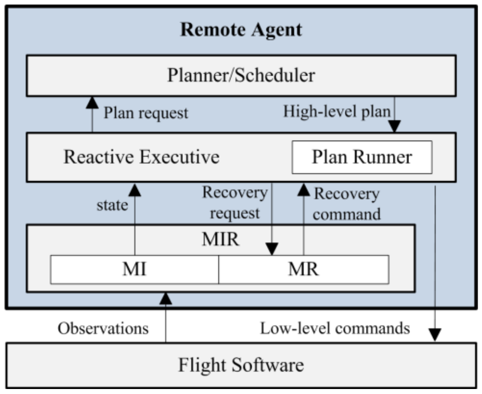


Figure 1: *The Architecture of Remote Agent* [*Muscettola et al. 1998*]

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.

IDEA, illustrated in Figure 2, is an agent-oriented architecture for distributed autonomy. Contrary to RA in which each layer has its internal concept representation, IDEA employs a unique concept representation for cooperating agents. The IDEA virtual machine is the core of the IDEA agent. It provides the deliberative and reactive planning and execution by employing reactive planner, plan database and plan runner. The employed components are similar to their counterpart components in RA architecture.

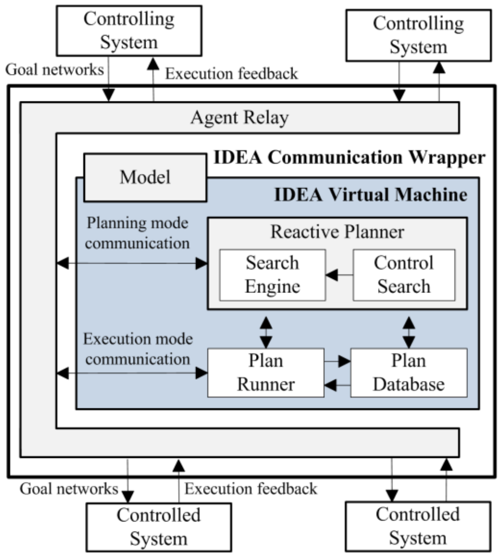


Figure 2: *The Architecture of IDEA* [*Muscettola et al. 2002*]

Structurally, the IDEA virtual machine integrates the planner/scheduler and reactive executive layers of the RA architecture. The communication wrapper provides and manages the communicational channels among the IDEA agents. The Model component shared between virtual machine and communication wrapper, sets the formal communication protocols among the agents. Each agent can control other agents or be controlled by them. In former case, the agent sends its goals in formal format to the agents that are being controlled and waits for their feedbacks, whereas in latter case, the agent receives the goals from controlling agents, plans and executes proper actions, and sends the execution feedback to the controlling agent.

3 Autonomous Migrating Agents

Schematic of our proposed architecture is shown in Figure 3. 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.

3.1 Cognitive Layer

Cognitive layer provides the agent 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. Concerning 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, 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.

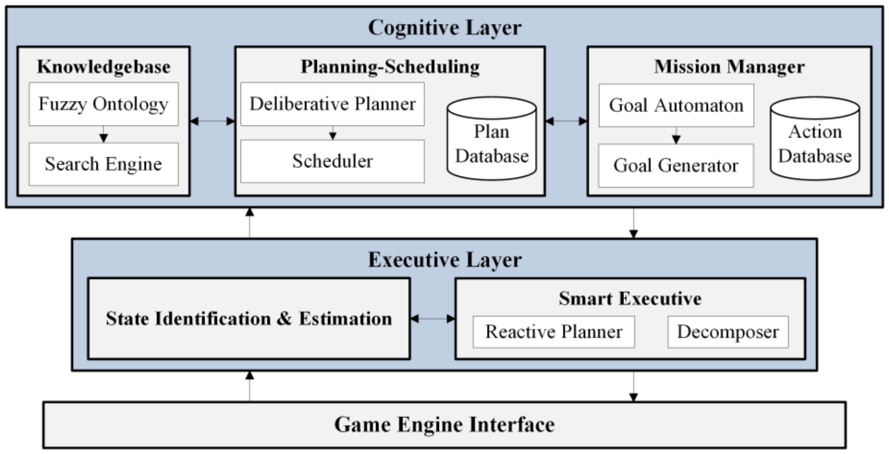


Figure 3: *The proposed autonomous architecture for migrating agents*

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 [Russell Norving 2010]. 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 [Hassani et al. 2013 (a)] 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 agent's virtual or physical 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.

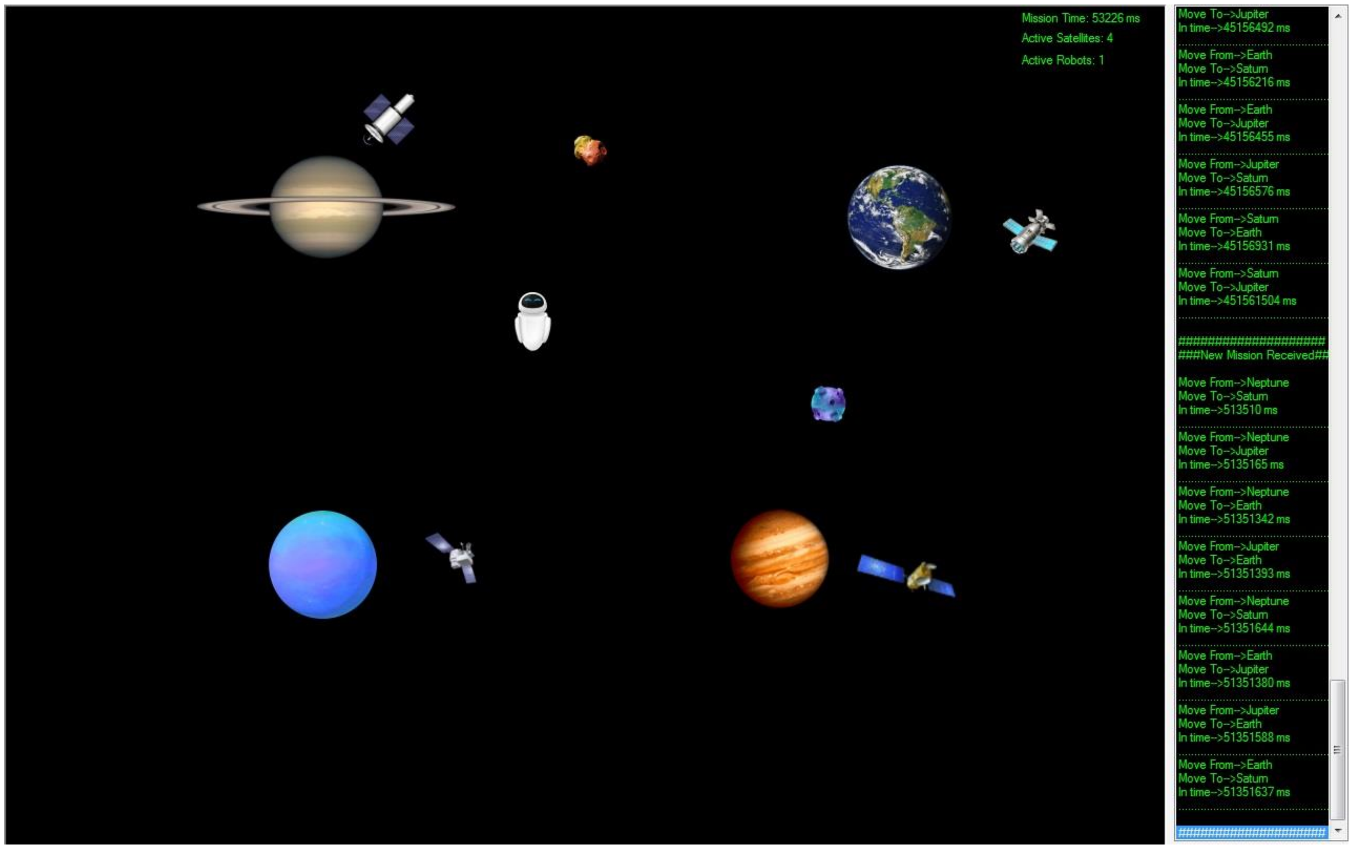
Smart executive is responsible for executing sequences of planned actions within plan database, and monitoring the execution process in order to prevent inconsistencies. It consists of two sub-components including decomposer and reactive planner. Decomposer fetches the scheduled action sequences from 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. It is noteworthy that this architecture can be tailored to different embodiments by defining associated actions and perceptions with target body in action database and state identification and estimation components, respectively.

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 interplanetary world. In this virtual world, we define four planets including Earth, Saturn, Neptune, and Jupiter. Also, we define four observatory satellites orbiting these planets with different velocities. These satellites orbit the planets either in a circular or an elliptic orbit. Moreover, two non-stationary asteroids with linear trajectory are embedded in the world. Ultimately, we created a robot avatar to serve as the embodiment of the proposed architecture.

The robot starts from a fixed position called origin and then non-periodically receives sequences of mission goals. The sequence contains some goal and their deadlines. Each goal consists of a source and destination satellites, and states that robot should pick up a payload from source satellite and transfer it to the destination satellite while satisfying the deadline and avoiding the asteroids and planets. The robot deliberatively plans and schedules for a complete sequence and executes the plans for whole mission. However, due to stochastic trajectories of asteroids and noise in satellite orbits, it utilizes its reactive planner to tweak the plans in real-time. The robot picks up the payload by setting a rendezvous with the source satellite and then moves to the destination satellite and sets another rendezvous with that satellite to deliver the payload. It is noteworthy that due to noisy estimations of satellite's trajectory and stochastic dynamics of asteroids, the world provides proper characteristics for evaluating both reactive and deliberative behaviors. A sample scene of the simulation is shown in Figure 4. The received mission goals containing information about source and destination satellites, and the temporal deadlines are displayed on mission panel on the left side of the simulations. The main simulation panel shows the dynamics of satellites, asteroids and the robot in soft real time. The agent and environment are implemented in C#.Net programming language.

The agent's knowledgebase contains the fixed positions of the planets, and the equations of dynamics of the position of asteroids and satellites. State identification and estimation unit monitors the trajectories computed based on these equations and in case of inconsistency due to random noise in the environment, it informs the reactive planner to correct the trajectory based on estimations from state identification and estimation unit. Thus, by defining a set of state variables, the proposed architecture can have an internal model of the universe it is operating in. It is noteworthy that by utilizing machine learning techniques that are devised for learning hidden variables such as EM algorithm, it is possible to omit the need for defining the state variables directly and let the agent itself to extract those variables. Moreover, regression and interpolation techniques can be utilized to learn the trajectories of dynamic objects within the world. Ultimately, agent can learn the optimal policies by exploiting reinforcement learning paradigms.



```
Mission Time: 53226 ms
Active Satellites: 4
Active Robots: 1
```

```
Move To-->Jupiter
In time-->45156492 ms
.....................
Move From-->Earth
Move To-->Saturn
In time-->45156216 ms
.....................
Move From-->Earth
Move To-->Jupiter
In time-->45156455 ms
.....................
Move From-->Jupiter
Move To-->Saturn
In time-->45156576 ms
.....................
Move From-->Saturn
Move To-->Earth
In time-->45156931 ms
.....................
Move From-->Saturn
Move To-->Jupiter
In time-->451561504 ms
.....................

######################
###New Mission Received##

Move From-->Neptune
Move To-->Saturn
In time-->513510 ms
.....................
Move From-->Neptune
Move To-->Jupiter
In time-->5135165 ms
.....................
Move From-->Neptune
Move To-->Earth
In time-->51351342 ms
.....................
Move From-->Jupiter
Move To-->Earth
In time-->51351393 ms
.....................
Move From-->Neptune
Move To-->Saturn
In time-->51351644 ms
.....................
Move From-->Earth
Move To-->Jupiter
In time-->51351380 ms
.....................
Move From-->Jupiter
Move To-->Earth
In time-->51351588 ms
.....................
Move From-->Earth
Move To-->Saturn
In time-->51351637 ms
.....................

######################
```

Figure 4: *A sample scene of simulation environment*

Three actions are explicitly defined in action database. These actions include setting rendezvous, moving payload, and avoiding asteroids and are shown in Table 1, 2, and 3, respectively. In these tables, preconditions, effects and sub-actions of each aforementioned action is elaborated.

Using these three feasible actions, cognitive layer is able to plan and schedule series of actions that can satisfy the mission constraints. We define two virtual sensors to provide the required perceptions. One is the position sensor that indicates the current position of the robot. The other one is a virtual vision sensor with a specified sight radius that lets the robot to sense its surroundings. Using these three actions and two perceptions, the proposed agent architecture is able to adapt to the simulated environment.

Table 1: *Action of setting rendezvous*

```
Action identifier: SetRendezvous(Satellite)
Preconditions:
    Position: close to orbit
Effects:
    Change in payload
Estimated execution time: T_Rendezvous
Sub-Actions:
    Status: Moving
    Point←EstimatePoint(Satellite)
    MoveTo (Point)
    Stop ()
    Status: Fixed
```

Table 2: *Action of moving payload*

```
Action identifier: MovePayload(Satellite src, Satellite des)
Preconditions:
    Mission: Available
    Deadline not passed
Effects:
    Change in position
Estimated execution time: #Steps
Sub-Actions:
    Status: Moving
    Find Path (src,des)
    Move ()
    Status: Idle
```

Table 3: *Action of avoiding asteroids*

```
Action identifier: AvoidAsteroid()
Preconditions:
    Asteroid in sight
Effects:
    Change in direction
Estimated execution time: T_changeDirection
Sub-Actions:
    If (Collision is estimated)
        Status: Moving
        ChangeDirection()
        RefinePlans ()
    Status: Moving
```

We run the simulations for 1000 times. In each simulation, the length of a mission sequences is selected by a random uniform distribution in the range of [5,25]. Also, the randomness of the environment (i.e. noise in estimating the trajectory of satellites and asteroids) is set to the range of [0,1] with the step size of 0.05. The first step in evaluating the proposed architecture is to validate the coordination and interactions among different units. To do so, we utilize timing diagrams to analyze the activation and deactivation of units in response to different scenarios. A sample timing diagram of the simulations is shown in Figure 5 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 5, components function in different frequencies. As an instance, because deliberative planner and scheduler unit belongs to a layer with more high-level activities in comparison with state identification and estimation unit, it has lower activation and deactivation frequency. Different timing diagrams from different scenarios suggest that there are proper interactions among the units.

In Figure 6, average deliberative and reactive behaviors of the agent in response to different length of mission sequences is illustrated. In this diagram, the environmental randomness is set to 0.3. As shown, the average number of activation of deliberative and reactive planners are proportional to the length of mission sequence. It is noteworthy that in an environment without any randomness, average number of deliberative planning for each mission sequence regardless of its length will be one (i.e. deliberative planner will plan only once for each mission sequence), and the average number of activations of reactive planner will be zero. On the other hand, as shown in Figure 6, with the randomness degree of 0.3 (i.e. 30% noise), number of unpredicted situations increases proportional to the length of mission sequence. However as depicted, the agent is able to deliberate and react, properly. Figure 7 illustrates the average number of reactions of reactive planner in order to avoid the asteroids and modify the predicted rendezvous coordinates. In this scenario, the mission length is set to four and the robot has to meet all the satellites. As it is shown, with the randomness of 0.5, the reactive planner reacts approximately twice which statistically is valid. Moreover, as depicted in Figure 7, average number of reactive rendezvous modifications tends to grow faster than average number of obstacle avoidance. This is because robot has to meet all the satellites, whereas due to existence of only two asteroids, robot may never see an asteroid on its path. Thus, there is always more chance that robot has to modify its rendezvous coordination in comparison with avoiding the asteroids.

As a conclusion, evaluation results suggest that the agent is able to perform both deliberative and reactive behaviors in a way that it can reach the mission goals. Also, the results suggest that the combination of deliberative and reactive behaviors is approximately optimal.

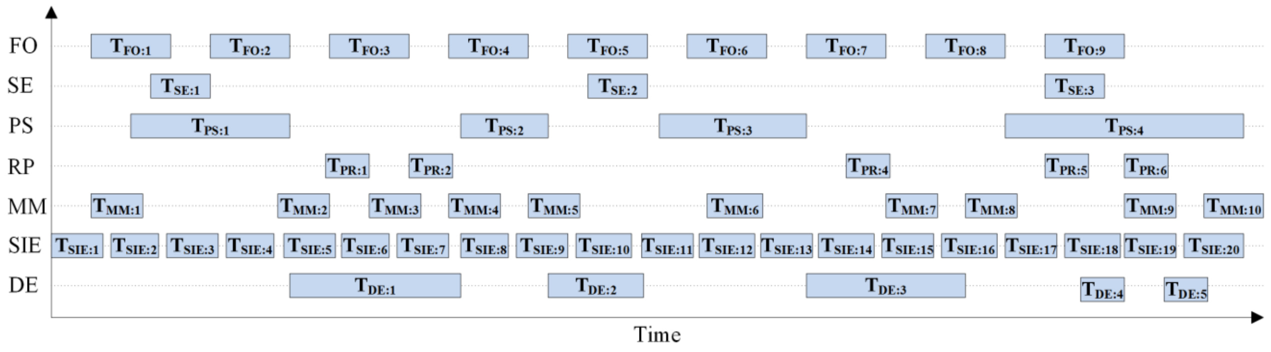

FO
SE
PS
RP
MM
SIE
DE
Time
$T_{FO:1}$ $T_{FO:2}$ $T_{FO:3}$ $T_{FO:4}$ $T_{FO:5}$ $T_{FO:6}$ $T_{FO:7}$ $T_{FO:8}$ $T_{FO:9}$
$T_{SE:1}$ $T_{SE:2}$ $T_{SE:3}$
$T_{PS:1}$ $T_{PS:2}$ $T_{PS:3}$ $T_{PS:4}$
$T_{PR:1}$ $T_{PR:2}$ $T_{PR:4}$ $T_{PR:5}$ $T_{PR:6}$
$T_{MM:1}$ $T_{MM:2}$ $T_{MM:3}$ $T_{MM:4}$ $T_{MM:5}$ $T_{MM:6}$ $T_{MM:7}$ $T_{MM:8}$ $T_{MM:9}$ $T_{MM:10}$
$T_{SIE:1}$ $T_{SIE:2}$ $T_{SIE:3}$ $T_{SIE:4}$ $T_{SIE:5}$ $T_{SIE:6}$ $T_{SIE:7}$ $T_{SIE:8}$ $T_{SIE:9}$ $T_{SIE:10}$ $T_{SIE:11}$ $T_{SIE:12}$ $T_{SIE:13}$ $T_{SIE:14}$ $T_{SIE:15}$ $T_{SIE:16}$ $T_{SIE:17}$ $T_{SIE:18}$ $T_{SIE:19}$ $T_{SIE:20}$
$T_{DE:1}$ $T_{DE:2}$ $T_{DE:3}$ $T_{DE:4}$ $T_{DE:5}$

Figure 5: *A sample timing diagram of the simulations*

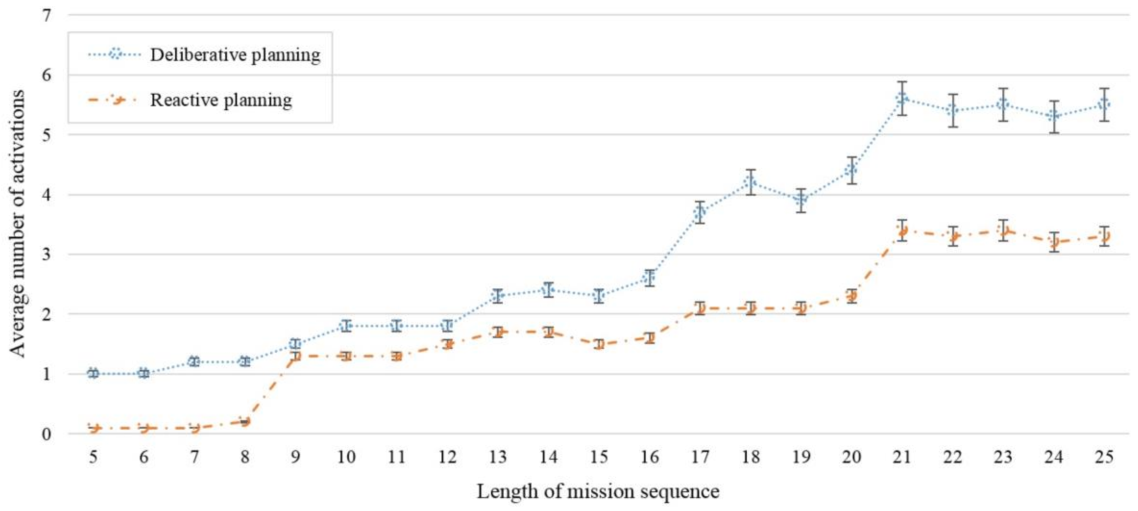


Figure 6: *Reactive and deliberative behaviors of the agent*

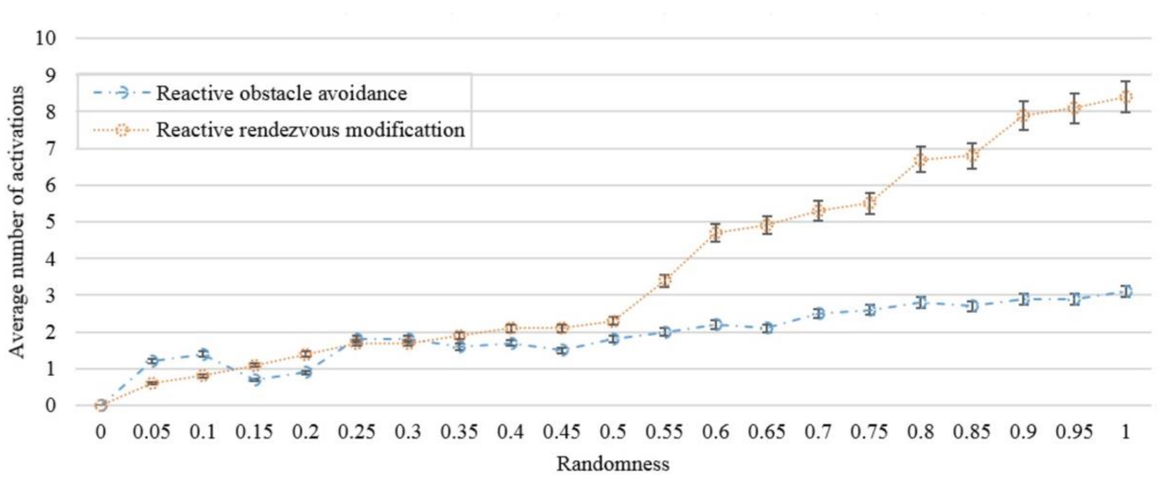


Figure 7: *Reactive behavior of the agent*

5 Conclusion

In the terminology of multi-agent systems, migrating refers to the ability of an agent to morph from one embodiment to another. In this paper, we introduced a generic architecture for developing autonomous migrating agents inspired by RA and IDEA agent architectures that are utilized in inter-planetary space missions for providing onboard autonomy. Our proposed architecture provides the agent with concurrent deliberative and reactive behaviors. The proposed framework lets the developers to tailor the embodiment by defining set of possible actions and perceptions associated with the new body. Furthermore, the proposed architecture 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 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. Ultimately, the simulations suggest that by defining a concise set of feasible actions, perceptions and state variables that agent can perform, percept and represent required knowledge, respectively, agent can move along different embodiments.

Currently, we are considering four directions for extending this research. The first direction is to utilize a mechanical embodiment beside the graphical one. In order to do that, we are planning to utilize a simple humanoid robot to perform the same tasks as its avatar counterpart does. Furthermore, we are planning to apply our agent architecture to a complex game scenario and investigate it is capability in playing the game autonomously. The third direction in this research is to perform a comprehensive user studies regarding the identity morphing of the proposed architecture. Ultimately, we are planning to exploit online machine learning approaches such as EM algorithm for learning hidden state variables, regression and interpolation techniques for learning the trajectories of dynamic objects within the world, reinforcement learning techniques for learning the optimal policies, and self-organizing neuro-fuzzy approaches for learning the ontology automatically from perception sequence. Using these learning schemes renders the need for expert's knowledge obsolete.

References

Alami, R., Chautila, R., Fleury, S., Ghallab, M., and Ingrand, F. 1998. An Architecture for Autonomy. *International Journal of Robotics Research*, 17, 4, 315–337.

Alt, J.K., Francisco, B., and Darken, C.J. 2011. A practical situation based agent architecture for social simulations. In *2011 IEEE 1st International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support*, 305–312.

Arent, K., and Kreczmer, B. 2013. Identity of a companion, migrating between robots without common communication modalities: Initial results of VHRI study. In *Proceedings of the 18th International Conference on Methods and Models in Automation and Robotics*, 109–114.

Arnellos, A., Vosinakis, S., Anastasakis, G., and Darzentas, J. 2008. Autonomy in virtual agents: Integrating perception and action on functionally grounded representations. In *Proceedings of the 5th Hellenic Conference on AI*, 51–63.

Aylett, R., Kriegel, M., Wallace, I., Marquez Segura, E., Mecurio, J., Nylander, S., and Vargas, P. 2013. Do I remember you? Memory and identity in multiple embodiments. In *Proceedings of IEEE RO-MAN conference*, 143–148

Barella, A., Carrascosa, C. and Botti, V. 2007. Agent architectures for intelligent virtual environments. In *IEEE/ACM International Conference on Intelligent Agent Technology*, 532–535.

Barriga, S.D., Rodriguez, L. and Ramos, F. 2012. Emotional attention in autonomous agents: a biologically inspired model. In *International Conference on Cyberworlds*, 61–68.

Bordini, R.F., Hübner, J.F. and Wooldridge, M. 2007. Programming Multi-Agent Systems in AgentSpeak Using Jason, *Wiley*.

Cassell, J., Sullivan, J., Prevost, S. and Churchill. E. 2000. Embodied Conversational Agents, *MIT Press*.

Conde, T., Tambellini, W. A and ND Thalmann, D. 2003. Behavioral animation of autonomous virtual agents helped by reinforcement learning. In *Proceedings of the 4th International Workshop on Intelligent Virtual Agents*, 175–180.

Conde, T., and Thalmann, D. 2005. Autonomous virtual agents learning a cognitive model and evolving. In *Proceedings of 5th International Conference on Intelligent Virtual Agents*, 88–98.

Conde, T., and Thalmann, D. 2006. An Integrated Perception for Autonomous Virtual Agents: Active and Predictive Perception. *Computer Animation and Virtual Worlds*, 17, 3, 457–468.

Conforth, M., and Meng, Y. 2011. CHARISMA: A context hierarchy-based cognitive architecture for self-motivated social agents. In

2011 International Joint Conference on Neural Networks, 1894–1901.

Dastani, M. 2008. 2APL: A Practical Agent Programming Language. *Autonomous Agents and Multi-Agent Systems*, 16, 3, 214–248.

Edward, L., Lourdeaux, D., and Barthes, J. 2009. An action selection architecture for autonomous virtual agents. In *New Challenges in Computational Collective Intelligence*, N.T. Nguyen, R.P. Katarzyniak, A. Janiak, Ed., Springer, 269–280.

Fong, T., Nourbakhsh, I., and Dautenhahn, K. 2002. A survey of socially interactive robots: Concepts, design, and applications. *Robotics and Autonomous Systems*, 42, 142–166.

Gomes, P., Segura, E., Cramer, H., Paiva, T., Paiva, A., Holmquist, L. 2011. ViPleo and PhyPleo: artificial pet with two embodiments. In *Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology*. No. 3.

Hassani, K., and Lee, W-S. 2013. A software-in-the-loop simulation of an intelligent microsatellite within a virtual environment. In *Proceedings of IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications*, 31–36.

Hassani, K., Nahvi, A., and Ahmadi, A. 2013 (a). Architectural Design and Implementation of Intelligent Embodied Conversational Agents Using Fuzzy Knowledge Base. *Journal of Intelligent and Fuzzy System*, 25, 3, 811–823.

Hassani, K., Nahvi, A., and Ahmadi, A. 2013 (b). Design and Implementation of an Intelligent Virtual Environment for Improving Speaking and Listening Skills. *Interactive Learning Environments*, 1–20. DOI:10.1080/10494820.2013.846265

Heudin, J. 2008. Evolutionary virtual agent at an exhibition. In *13th International Conference on Virtual Systems and Multimedia*, 154–165.

Hindriks, K.V. 2009. Programming rational agents in GOAL. In *Multi-Agent Programming: Languages, Tools and Applications*, A. Seghrouchni, J. Dix, M. Dastani, Ed., Springer, 119–157.

Horvath, G., Ingham, M., Chung, S., Martin, O., and Williams, B. 2006. Practical application of model-based programming and state-based architecture to space missions. In *Proceedings of the 2nd IEEE International Conference on Space Mission Challenges for Information Technology*, 80–88.

Kriegel, M., Aylett, R., Cuba, P., Vala, M., and Paiva, A. 2011. Robots meet IVAs: A mind-body interface for migrating artificial intelligent agents. In *Proceedings of the 10th International Conference on Intelligent Virtual Agents*, 282–295.

Langley, P., Laird, J.E., and Rogers, S. 2009. Cognitive Architectures: Research Issues and Challenges. *Cognitive Systems Research*, 10, 141–160.

Liu, Z., Hong, Y., Liu, Q., and Chai, Y. 2011. An emotion model for virtual agents with evolvable motivation. In *Transactions on Edutainment VI*, Z. Pan, A.D. Cheok, W. Müller, Ed., Springer, 154–163.

Liu, J., and Lu, Y. 2006. Agent architecture suitable for simulation of virtual human intelligence. In *the 6th World Congress on Intelligent Control and Automation*, 2521–2525.

Luengo, F. and Iglesias, A. 2003. A new architecture for simulating the behavior of virtual agents. In *Proceedings of the International Conference on Computational Science*, 935–946.

Luengo, F., and Iglesias, A. 2005. Designing an action selection engine for behavioral animation of intelligent virtual agents. In *Proceedings of the International Conference on Computational Science and Its Applications*, 1157–1166.

Mori, M. 2012. The uncanny valley. *IEEE Robotics & Automation Magazine*. 19, 2, 98–100.

Muscettola, N., Dorais, G., Fry, C., Levinson, R., and Plaunt, C. 2002. IDEA: Planning at the core of autonomous reactive agents. In *Proceedings of the 3rd International NASA Workshop on Planning and Scheduling for Space*.

Muscettola, N., Nayak, P., Pell, B., and Williams, B. 1998. Remote Agent: To Boldly Go Where No AI System Has Gone Before. *Artificial Intelligence*, 103, 5–48.

Nesnas, I., Simmons, R., Gaines, D., Kunz, C., Calderon, A., Estlin, T., Madison, R., Guineau, J., McHenry, M., Shu, I., and Apfelbaum, D. 2006. CLARAty: Challenges and steps toward reusable robotic software. *International Journal of Advance Robotic Systems*, 3, 1, 23–30.

Oijen, J., Vanhee, L., and Dignum, F. 2012. CIGA: A middleware for intelligent agents in virtual environments. In *International Workshop on Agents for Educational Games and Simulations*, 22–37.

Ortony, A. 2003. On making believable emotional agents believable. In *Emotions in Humans and Aartifacts*, R. Trappl, P. Petta, S. Payr, Ed., MIT Press, 189–211.

Pokahr, A., Braubach, L., and Lamersdorf, W. 2005. Jadex: A BDI Reasoning Engine. In *Multi-Agent Programming: Languages, Platforms and Applications*, R.H. Bordini, M. Dastani, J. Dix, A. Seghrouchni, Ed., 149–174.

Polle, B. 2002. *Autonomy requirement and technologies for future constellation*. Technical Report 3682899.02, Astrium Inc.

Rickel, J. 2001. Intelligent virtual agents for education and training: Opportunities and challenges. In *the 3rd International Workshop on Intelligent Virtual Agents*, 15–22.

Rumbell, T., Barnden, J., Denham, S., and Wennekers, T. (2012). Emotions in Autonomous Agents: Comparative Analysis of Mechanisms and Functions. *Autonomous Agents and Multi-Agent Systems*, 25, 1, 1–45.

Russell, S., and Norving, P. 2010. Artificial Intelligence: A Modern Approach (3rd ed.), *Prentice Hall*.

Sandamirskaya, Y., Richtert, M., and Schoner, G. 2011. A neural-dynamic architecture for behavioral organization of an embodied agent. In *IEEE International Conference on Development and Learning*, 1–7.

Segura, E., Kriegel, M., Aylett, R., Deshmukh, A., Cramer, H. 2012. How do you like me in this: User embodiment preferences for companion agents. In *Proceedings of 12th International Conference on Intelligent Virtual Agents*. 112–125.

Siegwart, R., Nourbakhsh, I.R., and Scaramuzza, D. 2011. *Introduction to autonomous mobile robots*. MIT Press, Cambridge.

Spinola, J., and Ricardo, I. 2012. A cognitive social agent architecture for cooperation in social simulations. In *Proceedings of the 12th International Conference on Intelligent Virtual Agents*, 311–318.

Steels, L., and Brooks, R. 1993. *The artificial life route to artificial intelligence: Building situated embodied agents*. Lawrence Erlbaum Associates, New Haven.

Stroupe, A., Singh, S., Simmons, R., Smith, T., Tompkins, P., Verma, V., Vitti-Lyons, R., and Wagner, M.D. 2000. *Technology for autonomous space systems*. Technical Report CMU-RI-TR-00-02, Carnegie Mellon University.

Thalmann, D. 2004. Control and autonomy for intelligent virtual agent behavior. In *Proceedings of the 3rd Hellenic Conference on AI*, 515–524.

Wooldridge, M. 2002. Intelligent agents: The key concepts. In *Multi-Agent Systems and Applications II*, V. Marik, O. Stepankova, H. Krautwurmova, M. Luck, Ed., Springer, 3–43.