A Software-in-the-Loop Simulation of an Intelligent MicroSatellite within a Virtual Environment

Kaveh Hassani and Won-Sook Lee
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Canada
kaveh.hassani@uottaw.ca and wslee@uottawa.ca

***Abstract*— Rapid growth in space missions necessitates the onboard intelligence, which creates autonomous space systems by providing high level decision making, robust execution of decisions, and automatic fault repairing. Mostly, autonomous space systems are implemented as hybrid architectures with a few conceptual layers. Validating the stability and evaluating the performance of an autonomous architecture is critical for space missions. Software−in−the−loop simulation is a suitable approach for addressing this demand. However, the data acquired from simulation is represented as alphanumeric values or diagrams, which needs to be interpreted. In this paper, we propose an intelligent architecture to provide onboard autonomy for an observation micro-satellite. The architecture integrates the low level physical actions with conceptual decision making ability in a hierarchical manner. To evaluate the proposed architecture, we have implemented a distributed software−in−the−loop simulation to simulate the space, satellite, ground stations, and intelligent onboard software. Moreover, for the first time, we have used virtual reality to visualize the satellite's autonomous behavior in the orbit. It lets the users have a high level feedback from integrated simulation. Scenario-based evaluations have shown the stability and efficiency of the proposed architecture.**

Keywords—Hybrid intelligent system; onboard autonomy; software-in-the-loop simulation; satellite visualization

I. INTRODUCTION

Since the beginning of space age in 1957, space technology has been progressing rapidly. Nowadays, satellite constellation systems such as GPS are serving individuals in daily life; interplanetary missions such as curiosity mars rover are transmitting rich information about other planets; and observation satellites are providing high resolution images for scientists. Space is an extreme environment which demands precise, efficient and robust technology [1]. Early space missions were proceeding under the ground team's direct supervision. A large crew was responsible for preparing low level instructions to satisfy a few mission goals. These instructions were uplinked to the satellite when it was passing over the ground station. However, due to the limited bandwidth and the low frequency of over passes, the satellite was mostly idle. Moreover, the satellite was not able to handle the unpredicted situations and had to be recovered manually by ground team. Another disadvantage was lack of ability to automatically detect and react to events such as natural phenomenon. High crew costs, mission inflexibility, lack of onboard problem solving ability, and limitations in mission definition are other drawbacks of conventional space systems.

Autonomous space technology emerged to alleviate the flaws of classic space systems. Polle has classified the motivations for autonomy in space technology to: cost reduction, improvement of the quality and quantity of mission products, availability, performance improvement, and new mission feasibility [2]. Dehgan, Cheheltani, and Kiapasha have mentioned the advantages of autonomous space systems as omitting time consuming activities by human, reliability enhancement by continuous system health monitoring, protecting system against its internal issues, and high ratio of useful data [3]. Onboard autonomy refers to the ability of performing a set of onboard activities to reach the mission goals in optimal configuration. In this paradigm, instead of low level instructions, high level missions are transmitted to the satellite. The intelligent onboard software plans, schedules, and executes the activities to achieve the mission goals with optimal resource allocation. Also, it monitors and reacts to the internal and external events. In case of none-fatal failures, it detects, isolates, and repairs the error. In summary, onboard autonomy pursues two main objectives: achieving high level goals and reacting in real-time [1].

Validating the stability and evaluating the performance of an autonomous architecture is critical for space missions. Software−in−the−loop (SIL) simulation is a suitable approach for addressing this demand. In SIL simulation, part of the model exists in standard simulation tool such as Simulink and the rest is compiled code [4]. However, the data acquired from simulation are alphanumeric values or diagrams, and need to be interpreted. A suitable approach for evaluating autonomous space systems is integrating virtual reality with SIL simulation to visualize the satellite behavior under the control of onboard intelligent software.

In this paper, we propose an intelligent architecture to provide onboard autonomy for an observation micro-satellite. The architecture integrates the low level physical actions with conceptual decision making ability in a hierarchical manner. To evaluate the proposed architecture, we have implemented a distributed SIL to simulate the space, satellite, ground stations,

and intelligent onboard software. Moreover, for the first time, we have used virtual reality to visualize the satellite's autonomous behavior in orbit. It lets the users have a high level feedback from integrated simulation.

The paper is organized as follows: in section 2 an overview of related works is presented; while Section 3 presents the flight scenario for the observation microsatellite. In section 4, the proposed intelligent architecture is described. In section 5, the integrated SIL simulation and visualization of the system is discussed. Finally, section 6 concludes the paper.

II. RELATED WORK

Autonomous architectures are originated from robotics. Early intelligent architectures were mostly based on sense-decide-act cycle [5] in which data is acquired from sensors and integrated with pre-defined goals to generate and execute a plan. Shakey was the first robot exploited this paradigm [6]. Although this paradigm is able to provide real-time responses, it cannot provide deliberative functionality. Stroupe et al. classified the autonomous architectures to three types: behavioral, hierarchical, and hybrid [1]. Behavioral architectures are bottom-up paradigms that model the system components as behaviors. This type is similar to sense-decide-act-based architectures and inherits the same problem. Brook's subsumption architecture is an example of behavioral architectures [7]. Hierarchical architectures are top-down approaches that decompose high level goals to low level sub-goals. Although they can handle the complex tasks, they are not efficient for tight reactive real-time tasks. RCS is an example of hierarchical architectures [8]. Hybrid architectures are layered architectures that combine deliberative features of hierarchical architectures with reactive real-time characteristics of behavioral architectures. LAAS [9] and Claraty [10] are examples of hybrid architectures. Remote agent (RA) is a hybrid architecture tested on deep space 1 [11]. It is designed to provide reliable autonomy for extended periods [12]. Intelligent distributed execution architecture (IDEA) is an agent oriented architecture that supports distributed autonomy [13]. It views each layer as an agent. IDEA has been evaluated on K9 rover [12]. Mission data system (MDS) is a hybrid software framework which emphasizes the state estimation and state control [14]. TITAN is another hybrid architecture that emphasizes model-based programming paradigm [14]. The modern space systems including satellite systems such as EO-1 [15] and Techsat-21 [16], and interplanetary missions such as DS1 [11] are exploiting hybrid architectures. Hybrid architectures are most efficient when coupled with hardware architectures such as GUARDS that address parallel processing and decomposition among layers [17].

Although many research works have been conducted on onboard autonomy, only a few have employed proper visualization methods. Some research works have exploited hardware simulations. Aghili, Namvar and Vukovich have mounted a satellite on a hydraulic manipulator to simulate a satellite in orbit [18]. Bodin, Nylund, and Battelino have implemented a hardware-in-the-loop (HIL) simulation to evaluate the onboard software in a simulated real-time environment [19]. Other research works have employed software modeling. Zhaowei, Guodong, Xiaohui, and Xibin have proposed an integrated system for design and simulation of a small satellite [20]. Miao, Chen, and Sun have proposed a distributed platform for satellite simulation [21]. A few research works have employed virtual environments for satellite simulation. A satellite simulator based on virtual reality is proposed in [22]. It visualizes the satellite orbiting the earth in 3D for training purposes. Hao, Pei, Yongkang, and Chao have integrated the 3D visualization and SIL for satellite simulation [23]. Another research work has been carried out to visualize the earth observing satellite within a virtual environment to facilitate the data interpretations [24]. In this paper, for the first time, we integrate virtual reality with SIL simulation to visualize the satellite behavior under the control of onboard intelligent software.

III. FLIGHT SCENARIO

The hypothetical mission is a remote sensing mission in which an observation microsatellite provides high resolution images from natural phenomena such as volcanoes for scientific analyses. The main components of the mission include an observation microsatellite with onboard autonomy; two unidirectional receiving ground stations; a Tracking, telemetry and command (TTC) station; and a mission control center (MCC). The mission components are shown in Fig. 1.

A. Microsatellite specification

The hypothetical satellite is a small scale observation satellite circling in low earth orbit (LEO). It is assumed that the microsatellite is already in the orbit and hence launching, releasing, and de−tumbling phases are ignored. The microsatellite includes four main subsystems. The payload subsystem is a panchromatic camera fixed to the satellite body. It is employed to take high resolution images from points of interest on the earth. The second subsystem is attitude determination and control system (ADCS). The satellite has 3+1 DOF (i.e. yaw, pitch, roll, and longitude), and cannot change orbit (altitude) or latitude. The ADCS is equipped with a GPS receiver, a biaxial solar tracker and an inertial measurement unit (IMU). Furthermore, it includes three reaction wheels each fixed along a rotational axis for closed-loop attitude control. The third subsystem is power provider which puts the satellite in an attitude with maximal solar energy absorption. The communication subsystem uplinks and downlinks data.

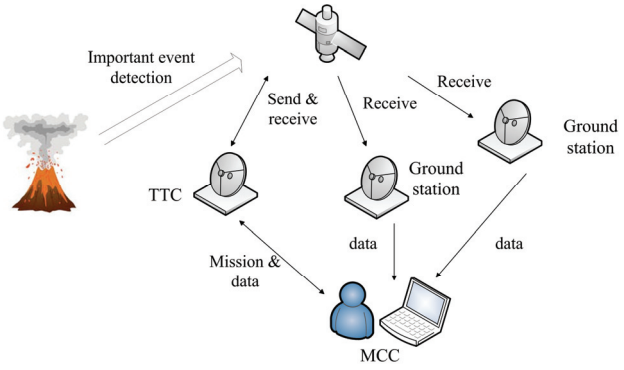

Fig. 1. The scenario of remote sensing mission.

B. Ground Stations

Four ground stations are considered: two unidirectional receiving ground stations, a TTC, and a MCC. The stations are connected via a virtual private network (VPN). The unidirectional ground stations receive the data and send it to MCC. TTC is a bidirectional station that monitors the performance of the satellite, tracks it in the orbit, and sends the logs to MCC. Also, it transmits prioritized missions from MCC to the satellite. MCC prioritizes the users' requests regarding parameters such as scientific importance. It generates high level missions and transmits them to TTC. Also, it synchronizes the ground stations and the satellite. The format of missions generated at MCC is depicted in (1).

$$IN\#LO\#LA\#NO\#FR\#AT\#PR \quad (1)$$

IN is the mission index; *LO* and *LA* are the longitude and latitude of the point of interest respectively. *NO* is the number of images to be taken from that point, *FR* is the imaging frequency, *AT* refers to the attitude mode (i.e. exact attitude pointing or Nadir attitude pointing), and *PR* indicates the mission priority. Using high frequency, the taken images will have small angular differences. This mode is useful for producing 3D images from the point of interest. On the other hand, having low frequency is proper to observe the dynamics of the point of interest. The payload is able to take images in both exact attitude and Nadir attitude pointing modes. In former mode, the given longitude and latitude must appear in the image center whereas in the later mode, appearance of the given coordinates inside the image suffices.

C. Scenarios

When the microsatellite passes over the TTC, the missions are uplinked to it. The onboard intelligent software plans and schedules some activities to complete uploaded missions based on priorities and resource constraints. As an example, when the satellite wants to take an image from a specific point, it first checks the power level. Then, it checks the availability of memory and camera. If these pre-conditions are satisfied, it determines the current attitude. Then, it changes the attitude using ADCS to aim to the point of interest. Finally, it takes the image. However, the scenarios are usually more complex. As another example, assume that the mission is to take three images from TTC and when the microsatellite is passing over TTC, the sun is in the view. In this scenario, satellite can employ all of the four subsystems. However, there is a contradiction between power providing and image taking subsystems. They both need to employ ADCS. The former subsystem wants to aim to the sun, and the later one wants to aim to the earth. Simultaneously, the satellite wants to communicate with TTC. Onboard intelligent software determines the sequence of activities to optimize the mission achievements with minimum cost.

The smallest entity used by onboard autonomy in decision making process is called action. An action represents an identical abstract activity to be performed by microsatellite. It encapsulates the low level details of an activity and provides a common knowledge representation between decision making and physical functionalities. Each action is defined by its pre-conditions, sub-actions, services, possible termination conditions, resource production and consumption, logs, estimated execution time, and effects. Pre-conditions of an action must be satisfied before it can begin. When an action is activated, it changes some states of the system, known as action effects. Sub-actions are functions executed in serial or parallel to map the action to low level instructions. As an example, attitude control is an action represented as follows:

Action: Attitude_Control
Estimated Time: 70s
Resources: Battery:0.05watt×T, CPU
Sub_Actions: Parallel(Sample(IMU,Solar)), Turn_On(Wheel), Control (Wheel), Turn_Off(Wheel)
Pre-conditions: IMU_STAT:idle, Solar_STAT:idle, Wheel_STAT:idle, Battery > α
Effects: Direction_STAT: Ready

IV. PROPOSED ARCHITECTURE FOR ONBOARD AUTONOMY

The proposed intelligent architecture, shown in Fig. 2, consists of three layers including decision layer, execution layer and functional layer.

A. Decision Layer

Decision layer plans and schedules the actions to reach the mission goals subject to resource constraints. It consists of two main components including mission manager (MM) and deliberative planer/scheduler (PS). MM prioritizes the missions using a weighted combination of priorities determined by ground team, resource constraints and spatial priorities. It uses a priority queue to sort and send the missions to PS. It includes two databases: action database and plan database. The action database keeps pre-defined actions and their attributes. Plan database contains pre-planned mission prepared by experts to boost up the planning process for routine missions. Before commencing the planning process, the mission is checked against this database. If any matches are found, the plans are fetched and directly scheduled.

PS exploits action database to search for the sequence of actions that reach the mission goals with minimum costs. It uses constraint-based iterative backtracking algorithm. PS determines the order of actions regarding the pre-conditions, effects, constraints and resource demands of actions. It incrementally adds actions to action sequence and in case that the pre-conditions are not satisfied, it backtracks and adds another action. Also, it uses data provided by state identification and estimation unit to determine the beginning time of the root action as lumped in (2).

$$T_s = T_c + T_e - \sum_{i=1}^{n} T_i \quad (2)$$

T_s is start time, T_c is current time, T_e is the time estimated to view the target, n is the number of actions to be executed, and T_i is an approximate execution time for each action.

PS is a deliberative module. It needs high processing time in scale of minutes, and once it has finished the process, the outputs suffice for a long temporal horizon in scale of hours.

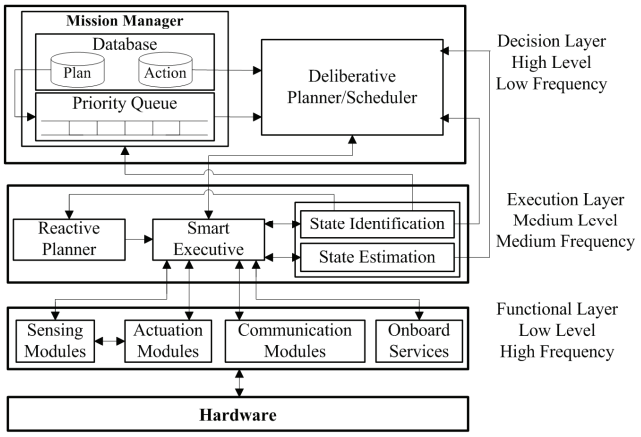

Fig. 2. Proposed autonomus architecture for onboard autonomy.

B. Execution Layer

This layer is a bridge between decision layer and flight software. It exploits the services in functional layer provided by flight software to execute the actions planed and scheduled by decision layer. Furthermore, it provides reactive responses in critical situations. Execution layer consists of reactive planner, smart executive and state identification and estimation unit (SIE).

Reactive planner responds to unpredicted situations. As mentioned before, PS plans for wide temporal horizons. However, accumulation of sensor errors, estimation errors, and unpredicted events can invalidate the plans. In this case, reactive planner responds to the situation using a heuristic algorithm. The heuristic algorithm uses the plan sequence and current state to perform an action to handle the situation in minimum cost. The cost is a linear combination of the mission failure cost and resource costs. In case of unrecoverable failure, reactive planner suspends the onboard autonomy and delivers the satellite control to the ground crew.

The smart executive receives scheduled actions from decision layer. Prior to determined execution time, it checks the pre-conditions of the action. If the pre-conditions are satisfied, it assigns a thread to the corresponding action and starts it. The threads are synchronized by semaphores [25]. Assigning threads to each action provides parallel execution. Moreover, the smart executive monitors the execution and detects inconsistencies. It monitors the estimated resource demands and the actual resource consumptions. If the difference exceeds the threshold, it terminates the action and informs the PS. If the failure is fatal (e.g. power level is very low), it activates reactive planner to transform the satellite to a safe mode. Also, when the plans are executed successfully, it informs the PS and demands new plans to execute. When an action thread starts, it invokes the sub-actions and executes them in serial or parallel. Each sub-action directly activates a hardware-oriented operation (i.e. sensing or actuating) or a service-oriented operation (i.e. calling image processing or data compression services) in functional layer.

SIE keeps the resource information (e.g. battery, memory, bandwidth etc.), mission information (e.g. ground station coordinates), reference timer, and updates them with high frequency. Furthermore, it assimilates data acquired from sensors using Kalman filter [26] and converts them to symbolic values (e.g. HIGH, OFF etc.) using fuzzifiers [27]. Moreover, it employs prediction functions to estimate the future states. These functions are dynamic orbital equations that get current state as input and calculate the next states. As an example, passing over function gets the current orbital coordinates, current reference time, and the target coordinates as input and computes the reference time in which satellite will pass over target coordinate. Using prediction functions, decision layer generates accurate plans for wide temporal horizons.

C. Functional Layer

Functional layer is an interface between software and hardware. It consists of modules which connect each sub-action to a particular hardware in a bijective manner. These modules can be categorized to four classes including: sensing modules, actuation modules, communication modules, and onboard services. Sensing modules are used to manage sensors. A sensing module activates a sensor, samples its data and deactivates it. The data from sensors are propagated to SIE via smart executive. Actuation modules are employed to control actuators. An actuation module consists of activation, deactivation and control sub-modules. Control sub-module can be a closed-loop controller (e.g. a digital PID controller for controlling a reaction wheel) or an open-loop controller which just sends specific signals (e.g. a shutter opening signal to payload). It is noteworthy that modules have local intercommunications to support real-time functionality. Also, it is noteworthy that those sensors that do not affect the decision making or execution process are hidden from upper layers. Shaft encoders used in reaction wheels are examples of these hidden sensors [28]. Communication modules include uplink and downlink modules which are employed to transmit data between satellite and ground stations. Onboard services are utility software such as image processing and data compression libraries. These services are invoked by actions to perform some data processing tasks.

V. Simulation and Experimental Results

In order to evaluate the stability and the performance of the proposed architecture, we have implemented an integrated SIL simulation and 3D visualization as illustrated in Fig. 3. SIL simulation provides accurate data from satellite's behavior in response to various missions and situations. However, this data is represented as alphanumeric values and 2D diagrams, and needs to be interpreted to provide high level knowledge. To do so, we employed 3D visualization to show the real-time feedback from satellite's behavior. It is noteworthy that we, for the first time, have integrated SIL simulations of flight software, onboard autonomy, and 3D visualization.

The simulation consists of onboard autonomy simulator, satellite simulator, environment simulator, ground station simulator and the communicator. The simulators perform a distributed SIL simulation. Each simulator runs on an identical computer with a 2.5 GHz processor and 1GB main memory.

The computers communicate with each other within a wireless local area network (LAN) using TCP/IP protocol.

Onboard autonomy simulator implements the decision layer and execution layer. This simulator is implemented using visual C#.Net 2008. Each layer is implemented as an identical process that communicates with its pair via a local socket. In other words, decision and execution layers are executed in parallel on a same computer. The communication protocol is TCP/IP.

The satellite simulator implements the functional layer and the satellite model. It simulates the virtual sensors such as GPS, sun tracker, star tracker, and IMU with high precision using STK 8 (satellite tool kit). Also, a linear model of actuators (e.g. reaction wheels) and the corresponding digital PID controller for real-time attitude control is implemented using visual C++.Net 2008. The functional layer and the satellite model communicate via a local port. The environment simulator employs dynamic orbital model to simulate LEO in which satellite is circling. It forces the satellite to follow the Newtonian mechanical rules. This simulator employs the encapsulated STK models to simulate the space environment.

Communicator is a middleware that manages the message passing between simulators. The messages contain data and header. Communicator uses headers to extract the destination simulator. It automatically converts the data embedded in message to the knowledge representation understandable by the target simulator. Finally, it composes a new message containing converted data as message body and the id of sender simulator as its header, and transmits it to the target simulator. The possible communication errors are automatically handled via TCP/IP exceptions. Furthermore, the communicator synchronizes the simulators by providing a common time reference. The simulators periodically synchronize their local time with the common time reference by sending a time request to the communicator. Communicator is implemented using visual C#.Net 2008.

The ground station simulator provides two types of user interface. The first interface simulates MCC. It let users to communicate with satellite by uplinking missions, receiving data from satellite, and monitoring the satellite's subsystems. This interface is implemented using visual C#.Net 2008 and employs windows graphical user interface (GUI) components. Although this interface displays data as alphanumeric values and visualizes them as curve plots in real-time, due to the high volume and update rate of data, it is not possible to interpret the data online. This interface is useful for monitoring the elaborated behavior of subsystems such as ADCS. Moreover, this interface creates rich logs from data flow and lets the crew analyze the preferred aspects of the satellite offline by using these logs.

The second interface complements the first interface. It visualizes the satellite's behavior in a 3D virtual environment and provides an interactive 3D view of satellite orbiting the earth. This interface gives a high level interpretation of system which is easily understandable by users.

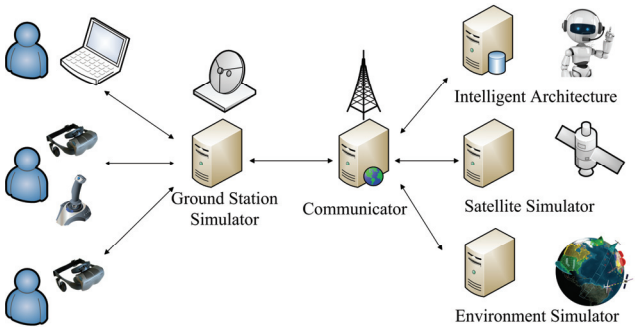

Fig. 3. Integrated software-in-the-loop simulation and visualization.

Using this interface, two users view the real-time simulation simultaneously by wearing two HMDs. Two Video 3D Pro i-glasses are employed as HMDs. Furthermore, one of the users is considered as the leader user who can interact with the environment by changing the point of view using an Extreme 3D Pro joystick. When the leader user changes the point of view, the follower user feels the changes concurrently. By employing this interface, the users monitor the satellite's behavior in real-time. Users perceive behaviors such as power providing (visualizing attitude changes toward the sun), imaging (visualizing attitude changes toward the target point and highlighting target point on earth), communicating (visualizing transmitted electromagnetic signals between satellite and ground station, and highlighting target point on earth) by viewing realistic physical behaviors of satellite in real-time in the space environment. Using this technic, users can primitively evaluate the stability and performance of the satellite and then confirm their observations using data logs from first interface.

To validate the stability and evaluate the performance of the proposed autonomous architecture, various missions are planned by onboard intelligent software. The experts are asked to plan the same missions too. Then, the planned missions prepared by both software and experts are executed on simulation framework and the results are compared as shown in Table I. Comparing the average number of actions planned by experts and intelligent software per mission shows that the intelligent software outperforms the experts by 30%. Also, comparison between average failures per mission shows that the intelligent software is 32% more reliable than experts. Moreover, considering the computed standard deviations, it can be inferred that the intelligent software performs better in terms of stability. It is noteworthy that by increasing the number of missions, efficiency and stability of experts' plans decrease rapidly, whereas these criteria decrease slightly for onboard software. A comparison between the characteristics of four hybrid architectures and our architecture is shown in Table II.

TABLE I. PERFORMANCE OF INTELLIGET SOFTWARE AND EXPERTS

Per mission	Intelligent software		Experts	
	Mean	***SD***	***Mean***	***SD***
Actions	4.37	1.45	6.23	3.84
Failures	0.08	0.03	0.25	0.17

TABLE II. COMPARISON BETWEEN THE ARCHITECTURES

Architecture	Methodology	#Abstraction	Mission type
IDEA	Muti-agent	Two levels	Constellation
RA	Multi-layered	Three levels	Interplanetary
MDS	State-based	One level	Flight-Test
TITAN	Model-based	One level	Flight-Test
Proposed	Multi-layered	Three levels	LEO

VI. CONCLUSION AND FUTURE WORKS

In this paper, we proposed an intelligent architecture to provide onboard autonomy for an observation micro-satellite. The architecture integrates the low level physical actions with conceptual decision making ability in a hierarchical manner. To evaluate the proposed architecture, we implemented a distributed SIL simulation to simulate the space, satellite, ground stations, and intelligent onboard software. Moreover, for the first time, we used virtual reality to visualize the satellite's autonomous behavior in the orbit. It lets the users have a high level feedback from integrated simulation. Scenario-based evaluations showed that in average, the proposed architecture generates 30% less actions in comparison with experts per mission. Also, results show that the average failures per mission caused by onboard software are 32% less than experts. Finally, comparison between standard deviations shows that the onboard software shows more steady responds in comparison with experts' plans. Two developments are considered as future woks. First, the simulation will be integrated with Google earth to show realistic images taken from the point of interest. Second, the architecture will be modified to handle the distributed missions such as cooperating constellation of satellites and rendezvous planning.

ACKNOWLEDGMENT

Authors would like to thank the guidance, navigation and control group members, especially Dr. Dehgan, Mr. Karimian, Mr. Cheheltani, Mr. Gheibi, and Ms. Hemmatabadi.

REFERENCES

[1] A. Stroupe, S. Singh, R. Simmons, T. Smith, P. Tompkins, V.Verma, R. Vitti-Lyons, and M.D. Wagner, "Technology for autonomous space systems", Robot. Inst., Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU-RI-TR-00-02, 2000.

[2] B. Polle, "Autonomy requirement and technologies for future constellation" Astrium, Houston, TX, Tech. Rep. EAA.NT.BP. 3682899.02, 2002.

[3] S.M. Dehghan, S.H. Cheheltani, and A. Kiapasha, "Design and implementation of intelligent decision making system to generate flight scenario automatically", in *5th Int. Con. on Recent Advances in Space Technologies*, 2011, pp.178-183.

[4] B.S. El-Haik and A. Shaout, *Software Design for Six Sigma: A Roadmap for Excellence*. NJ:Wiley, 2010.

[5] R. Siegwart and I.R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*. Cambridge, MA: MIT Press, 2004.

[6] R.E. Fikes, P. E. Hart. and N.J. Nilsson, "Learning and executing generalized robot plans", *Artificial Intell.*, vol. 3, pp.251-288, 1972.

[7] R. Brooks, "A robust layered control system for a mobile robot" *IEEE Trans. Robot. Autom.*, vol. RA-2, pp.14-23, Mar. 1986.

[8] J. Albus and W. Rippey, "RCS: a reference model architecture for intelligent control", in Proc. *From Perception to Action*, 1994, pp.218-229.

[9] R. Alami, R. Chautila, S. Fleury, M. Ghallab, and F. Ingrand, "An architecture for autonomy" Int. J. of Robot. Research, vol. 17, no. 4, pp.315-337, Apr. 1998.

[10] I. Nesnas, R. Simmons, D. Gaines, C. Kunz, A. Calderon, T. Estlin, R. Madison, J. Guineau, M. McHenry, I. Shu, and D. Apfelbaum, "CLARAty: challenges and steps toward reusable robotic software", *Int. J. of Adv. Robot. Sys.*, vol. 3, no. 1, pp.23-30, 2006.

[11] N. Muscettola, P. Nayak, B. Pell, and B. Williams, "Remote agent: to boldly go where no AI system has gone before", *Artificial Intell.*, vol. 103, pp.5-48, 1998.

[12] M.S. Boddy, S.A. Harp, and K.S. Nelson, "CLOCKWORK: requirements definition and technology evaluation for robust, compiled autonomous spacecraft executives", NASA, CA, Tech. Rep. Grant NAG-2-1624, 2004.

[13] N. Muscettola, G.A. Dorais, C. Fry, R. Levinson, and Christian Plaunt, "IDEA: planning at the core of autonomous reactive agents", presented at the Proc. of the 3rd Int. NASA Workshop on Planning and Scheduling for Space, Houston, TX, 2002.

[14] G. Horvath, M. Ingham, S. Chung, O. Martin, and B. Williams, "Practical application of model-based programming and state-based architecture to space missions", in *Pro. of the 2nd IEEE Int. Con. on Space Mission Challenges for Information Technology*, 2006, pp.80-88.

[15] B. Cichy, S. Chien, S. Schaffer, D. Tran, G. Rabideau, and R. Sherwood, "Validating the autonomous EO-1 agent", presented at the Int. workshop on Planning and Scheduling for Space, Darmstadt, Germany, 2004.

[16] R. Sherwood, S. Chien, R. Castano, and G. Rabideau, "Autonomous planning and scheduling on the TechSat 21 mission", in *Proc. of the 15th Australian Joint Con. on Artificial Intelligence*, 2002, pp.213-224.

[17] D. Powell, J. Arlat, L.B. Dukic, A. Bondavalli, P. Coppola, A. Fantechi, E. Jenn, C. Rabejac, and A. Wellings, "GUARDS: a generic upgradable architecture for real-time dependable systems", *IEEE Trans. Parallel Distrib. Syst.*, vol. 10, no. 6, pp.580-599, Jun 1999.

[18] F. Aghili, M. Namvar, and G. Vukovich, "Satellite simulator with a hydraulic manipulator", *in Proc. of the 2006 IEEE Int. Con. on Robotics and Automation*, Orlando, pp. 3886-3892.

[19] P. Bodin, M. Nylund, and M. Battelino, "SATSIM—A real-time multi-satellite simulator for test and validation in formation flying projects", *Acta Astronautica*, vol. 74, pp.29-39, 2012.

[20] S. Zhaowei, X. Guodong, L. Xiaohui, and C. Xibin, "The integrated system for design, analysis, system simulation and evaluation of the small satellite", *Adv. in Eng. Softw.,* vol. 31, pp.437-443, 2000.

[21] Y. Miao, C. Chen, and Z. Sun, "A satellite system distributed simulation design and synchronous control", in *Proc. of the 2009 IEEE Int. Con. On Mechatronics and Automation*, pp.3889-3893.

[22] M. Mirshams, A. Nahvi, M. Khosrojerdi, H. Tae, and Ali Vahid, "A 6-DoF satellite virtual simulator design and development", *Appl. Mech. and Mater.*, vol. 186, pp.70-74, 2012.

[23] H. Hao, C. Pei, S. Yong, and H. Chao, "Real-time three dimensional simulation platform for satellite mission analysis", in *6th IEEE Con. on Industrial Electronics and Applications*, 2011, pp.2593-2598.

[24] R. Witt, M. Fritz, T. Kuwahara, A. Brandt, C. Laurel, H.-P. Roser, and J. Eickhoff, "Real-time 3D Visualization in Satellite Development", in *4th Int. Con. on Astrodynamics Tools and Techniques*, Madrid, Spane, 2010.

[25] A.S. Tanenbaum, *Modern Operating Systems*. 3rd ed. Upper Saddle River, NJ: Prentice Hall, 2008.

[26] G. Evensen, *Data Assimilation: The Ensemble Kalman Filter*. 2nd ed. Berlin, Germany: Springer, 2009.

[27] A. Kandel and G. Langholz, *Fuzzy Control Systems*. Boca Raton, FL:CRC Press, 1994.

[28] M.C. Chou, C.M. Liaw, S.B. Chein, F.H. Shieh, J.R. Tsai, and H.C. Chang, "Robust Current and Torque Controls for PMSM Driven Satellite Reaction Wheel", *IEEE Trans. Aerosp. Electron. Syst.*, vol. 47, no.1, pp. 58-74, Jan. 2012.