A generic multi-layer architecture for multi-spacecraft mission simulation

John M. Beighle
Air Force Institute of Technology Comp. Science Dept.
Air Force Institute of Technology
Dayton, Ohio
john.beighle@AFIT.edu

***Abstract*—The strategic advantages and known vulnerabilities of space technology infrastructure continues to gain momentum as hot topics in AI based research. International law has restricted space from adversarial aggression in this domain; however, increasing technological continuity concerns have many interested parties desiring insurance against potential cyber related risks to these systems. This research looks to address this interest in terms of offering a robust way to simulate various hypothetical threat scenarios with a generic and simple multi-layered architecture for autonomous, multi-agent, space-vehicle scenario modeling. It employs the popular video game Kerbal Space Program (KSP) as the simulator and explores the potential of other high-performance AI enablers like the Multi-Agent System (MAS) frameworks and Robot Frameworks (RF) like Robot Operating System v2 (ROS2). It expects to help shape conversations related to increasing delivery speed and quality of scenario development within the space simulation realm. It attempts a multi-agent use case scenario to assist determining the utility of the proposed multi-agent, distributed approach. The approach will show potential as the foundation for modelling complex, multi-agent or other machine learning (ML) AI scenarios if it is easy to use and extend.**

I. Introduction

As information technology evolves, modularized space systems enable more and more nations to develop and run their own space programs. These systems are increasingly becoming essential across the globe in support of broad arrays of public and private objectives [1, 2]. Their cyber profiles are of increasing concern. One of the primary challenges ensuring these system's continuity is the difficulty and costs associated with building and testing physical equipment in the real environment. Another is the high cost of building one-off simulators. Thus, modelling and simulation advances effective in these settings have no shortage of public and private enthusiasm [4].

To meet the increasing demand for fast-paced, multi-agent, cyberspace scenario development and to properly address the mentioned complexities, this work explores a new approach. It proposes the popular video game Kerbal Space Program (KSP) as the simulation environment for a multi-layered, distributed approach to multi-agent system development. It also briefly looks at the potential for integrating common multi-agent tools. KSP is a mature platform, in its 10th year of production, with well documented software development kits, external application programming interfaces (APIs) and advanced tutorials. It is highly extensible and enjoys an exceptionally large, active, developer support community focused on driving solutions and new ideas.

The remainder of the paper is structured as follows: Related Work describes associated domain related areas of development that are both historical and current. Background & Methodology visits surrounding topic considerations and defines the methodology in detail applied in this experiment, and Conclusions & Future work describes the results of the experiment and concludes.

II. Related Work

With the current administration in the U.S. pushing the pace of space defense interests it is more and more important for nation state's space agents to become more intelligent in dealing with complex choices in broader scenarios without the direct aid of humans. With traditional AI approaches there are two attractive options for addressing this quest: machine learning based approaches and multi-agent systems approaches [8]. The latter is the area this work is focused on and there are several preceding works contributing inspiration. The first is a Belief-Desire-Intention (BDI) model work that integrates BDI into the ROS2 framework [8]. This work adds interesting developments that leverage the real-time elements of ROS2 and uses it to allow agents to factor in time-based considerations and constraints when generating intentions and allows for more complex action schedules. The second, is the Unified Behavior Framework (UBF) [7] that leverages on-the-fly arrangements of simple agent behaviors and arbiter models to create desired reactions to varying external conditions [7]. This work will eventually move towards integrating, to some degree, one or both these inspiring ideas.

Another related work is taken from a manufacturing context where multi-agent systems are rapidly being applied to solve complex problems [9]. In this work social abilities are required for performing intelligent interaction (peer-to-peer negotiation and decision-making) among the different and heterogeneous Cyber Physical Production Systems (such as machines, transport systems and other equipment present in the factory) to achieve manufacturing reconfiguration [9]. This effort contributes a generic multi-layer architecture that integrates an RF with a MAS to provide social abilities to autonomous transport vehicles (ATVs). The framework introduced is implemented with ROS and Java Agent Development Environment (JADE). These two products are popular RF and MAS frameworks, respectively. They claim their implementation to be a first work addressing intelligent interaction of transportation systems for flexible manufacturing environments in a holistic form.

In terms of simulation, this research addresses a few capability gaps that exist in space-based, rapid-concept-development software tools in both NASA's and the AF's open source, and Government-Off-The-Shelf (GOTS) offerings including what is available in the Air Force's Institute of Technology (AFIT) and the Air Force Research Laboratories (AFRL) [5, 6, 7]. NASA's open source platform Ames Mission Control Tech (MCT) a real piece of mission software that is available to the public but not really a simulator [4]. The products available with the AF and AFIT are GOTS and therefore specialized for specific use cases.

III. Background & Methodology

A. *Hypothesis:*

This work looks to explore the limiting factors related to researchers who want to explore space-related AI research but may lack the expertise required to successfully jumpstart themselves with specialized GOTS products. Believing it is easier to self-start with larger, crowd-sourced type initiatives like KSP that has a large development audience and numerous on-going development initiatives, this work explores the potential to use KSP as an AI simulation tool, with advanced visualization, for rapidly conceiving and communicating AI concepts. This can hopefully assist in maximizing the evolutionary rate of development of AI practitioners who want to apply research topics to space [4].

B. *Background:*

There are numerous attractive elements of KSP for AI researchers looking for simulation data or a visualization tool. It implements a realistic space simulation physics engine like many modern space simulators [11]. The way in which its spacecraft are affected by planet's and other object's forces are driven by complex mathematics. The corresponding data, the very data AI needs to operate on, is not easy to generate. KSP sets itself apart here by making it easy to expose virtually any internal data point to external consumers through the Telemachus telemetry mod. When requirements come down to implement advanced ML, the use of ready-made tools for this exist in ROS2 to plug in and consume. ROS2 offers well-established methods for users to build upon each other's work by sharing reusable packages on their packages site, especially AI centered packages since focus is robotics. There are roughly six thousand different pluggable and open-source packages currently available. One of them is openai_ros which comes with robust tutorials. Future work will involve adding ML based AI as a compliment to the current multi-agent AI focus here. As mentioned previously, if the performance of the reusable energy-based satellite parts is encouraging, future work may focus on ML techniques applied to the design of endurance based, autonomous, micro satellites built with multi-objective, multi-agent strategies. Additionally, recharging/refueling station ideas can be looked at as a required time and defensive measures feasibilities for countering this type of AI cyber threat. It may be surprisingly difficult to counter. These future potentials made available via KSP's Telemachus and ROS2's combination make for exciting AI prospects. This streamlined approach to ML is another reason to consider the strength of KSP as a more serious AI based prospect.

The present primary challenge with KSP, that keeps it from being considered for use as a MAS front end, is the fact that it does not natively support multi-player. To control multiple satellites simultaneously, a 3rd party KSP add-in must be utilized which allows multiple disparate game instances to work as one through a multi-player server. Since this work explores the use of the most popular multi-player mod, Luna multi-player (LMP), the results will prove whether this is industrial enough to be used as a MAS enabler. Ideally, to overcome the distributed middleware complications of real MAS implementations and to abstract those layers, a formal MAS (JADE) framework would additionally be utilized on top of the middleware. However, at present, and with respect to our short implementation timeline, the MAS piece will be pushed into a future effort. For now, satellite telnet interfaces will be utilized to interface with and control each satellite's onboard central processing units (CPUs) and to enable the rudimentary, multi-layer, social behavior. The telnet interfaces perform as the distributed messaging and execution framework that tie all the satellites together.

Thus, using LMP and onboard telnet interfaces, the approach simulates very nicely a multi-player environment for KSP. Multi-agent satellites can be viewed on one instance but controlled from various other distributed instances. This simplistic approach, without ROS2 and JADE, is advised for this paper by colleagues to ease the introduction to the powerful multi-agent setting without having to also climb the steep learning curves associated with ROS2 and JADE environments. Another, more advanced, version of

this paper will include ROS2 for advanced ML and distributed middleware elements. In a complimentary fashion the ROS2 services layers will use the same telnet interfaces seen in this approach as wrappers to communicate to the discrete satellites. Additionally, ROS2 topics will be derived from the Telemachus' telemetry data. Future versions of this paper will include a formal MAS layer like JADE. We currently have the ROS2 Python service nodes up and running in the KSP Windows environment.

C. Method:

The method to create the multi-layered, distributed, multi-agent capability for automating and enabling AI in space, as discussed, is a simple recipe. Perhaps simple once seen and used but given web searches produce no results the method here indicates this is the first case, as far as the author knows, of KSP being leveraged as a distributed MAS core framework simulator. The basic method uses KSP, Kerbal Operating Script (KoS) for both scripting and its control hardware, including the craft's onboard telnet servers that are interfaced via wireless radio communication with boot scripts and real-time loadable software images, Luna multi-player and the Relatively Adequate Mission Planner suite. These are the core KSP mods that make it possible.

Each satellite is autonomous and has its own behavior which is enabled by onboard CPUs and launched either by the craft at launch or by a higher layered MAS controller. In our experiment, each craft is loaded with custom code and executed by the social layer controller at the onset of the mission. The code is a mission plan for each craft to perform a single mission. That is the extent of the use case. The mission can be set up to run multiple objectives or new code behavior can be loaded from the ground-station based social controller to set up new missions in real time if desired. One must tackle the KoS scripting language. It is straightforward, extensive, and professionally written and maintained with excellent online documentation and resources.

It is worth noting that all KSP mods used are actively being updated and have active development support. Also, all are free and open-sourced. The use case described below does not mention AI implementations related to getting the satellites into staged execution states. For instance, we forgo writing AIs for launching and flying the carrier vehicles from Earth and maneuvering satellites into proper orbits and positions. Placing crafts into staging position is easy within the game and the use case begins with satellites in relative positions to begin their mission.

The computer running the entire four-player orchestration and the LMP server is an i9-9900 CPU @ 3.60GHz, 64.0 GB RAM, NVIDIA Quadro P4000 graphics, running Windows 10. It is quite easy to run multiple instances of the game on the same machine. Merely make a copy of KSPs root directory some else and creating corresponding symbolic links to a global scripts directory to keep from having to run down code in multiple locations. The functional layer code sets up each craft's telnet addresses when they start thus enabling the social layer to communicate with numerous agents. The limit to LMP is somewhere around 20 concurrent agents at present.

D. Use Case:

This research utilizes the following use case to explore the overall characteristics and effectiveness of using KSP for multi-agent simulation. Two autonomous Red team satellites, each equipped with onboard functional agents, controlling movements, and coordinating with a social layer to ensure overall objective success. A social layer controls the highest-level decisions and sorts high level conflicts in a multi-layered control structure [9]. The overall objective for this team is to disable a Blue team's constellation of satellites one target (goal) at a time. It entails the Red team launching into Medium Earth Orbit (MEO, 20,000km, constellation altitude). The Red team will begin their flight mission to the first Blue team satellite goal to inspect it. Each goal satellite has its own protector micro-satellite. These Blue micro protectors will attempt to physically block adversaries from touching its protected goal satellite. The protectors are not overly advanced and can be overcome by more than one encroaching adversary. It will engage the first approaching threat and attempt to run it off. Thus, the job of one of the two Red satellites (decoy) is to preoccupy Blue's protector while the other attempts to approach from another direction to inspect/dock the goal satellite. For the purposes of this experiment, docking a satellite renders it successfully out-of-service, and under Red's control. After docking, the Red team can move onto the next member of the constellation for a sequential and continuous inspection that diminishes Red's mission service levels.

The Red team satellites will be simulating renewable energy-based propulsion to assist development of baseline times and fuel requirements to perform multi-target maneuvers to better understand basic practical range and limitations of micro satellites leveraging reusable energy-based navigation. Ultimately, if the results are encouraging, future work may employ deep reinforcement learning (DRL) to deliver feasibility studies for design of endurance based, autonomous, micro satellites built with multi-objective, multi-agent strategies in mind. Additionally, recharging/refueling station ideas can be looked at also as a required time and defensive measures feasibilities for countering this type of AI based potential. The employment here of reusable energy concepts, Near Future Electrical and Near Future Propulsion engines, thrusters, batteries, and devices to navigate, create an exciting opportunity for researching potential deep space advantages.

The orbital mechanics that Red will use to inspect Blue's constellation is foundationally based on concepts outlined in Rocket & Space Technology resources [10]. We are applying predominately a mixture of common maneuvering and hill-climbing search algorithms to effectively navigate Bs to their correct inclinations and targets. Within the broader navigations flight control is simply leader-follower proximity reactive control systems (RCS) ensuring a collision avoidance throughout target approaches and dog fighting engagement. Once within target proximity (400 meters) the team begins execution of fixating Blue's micro protector on one Red target while the other Red micro attempts inspection. This requires Blue protectors to be equipped with onboard navigational AI

that will engage the threat and continually navigate itself between the threat and the satellite its protecting. It will automatically force a collision and self-detonate if a threat does not respond and move away from the goal.

The simplistic MAS design begins with only functional controllers that are directly attached to the agents themselves (the satellites) and the ROS2 "social layer" node which is the overall "mission agent" providing overarching and final decision arbitration.

1) Social Layer

The ground station based top layer controller is the social layer of this simplistic and custom MAS design. This is also the agent responsible for providing coordination for the entire mission. Its job is to receive satellite events and data from the functional controllers and drive the mission intelligently through to completion with high level orders. In this sense it plays the role of Mission Controller within the operative layer of a traditional MAS construct and covers the role of a traditional cognitive layer [9].

2) KSP Satellite Agent

Satellite agent autonomy is realized in KSP by designing a remote-controllable spacecraft that has an onboard "Scriptable Control System" component like the CX-4181 externally mountable computer part from the kOS add-in. Once this is attached the craft becomes autonomously controllable and onboard kOS scripts can be executed either remotely or via launch sequences. Additionally, with the kOS telnet feature enabled and Telemachus TeleBlade Antenna added to the craft, it is now capable of two-way remote communication data exchange and execution control.

IV. Conclusion & Future Work

The results of this work and the use case can be verified with this paper's corresponding video demonstration. The concept of using KSP as a core component in a MAS, distributed framework, remains a feasible idea given the intent. LMP is constrained by 50 max players so it is suitable for medium but not large scenarios. Using KOS is fast and efficient within the game and code changes happen very quickly. The experience of using KSP is straight forward and stable in terms of how vastly modable it is. LMP is easy and stable in most all cases. The expanse of potential real-time simulation data that can be made easily available to external consumers through the Telemachus module and consumed over ROS2 topic nodes presents serious potential for AI enthusiast. This work is the beginning of either incorporating a formal MAS like the JACK agent framework, which specializes in agent planning, or JADE which offers belief, desire, intent (BDI) based enhanced real-time planning and messaging abstraction. Also, using ROS2 for bagging Telemachus data and employing Openai_ros for future ML requirements will provide for exciting future research.

References

[1] Defense Intelligence Agency, "Challenges To Security in Space," *Mil. Power Publ.*, no. January, p. 46, 2019.

[2] N. Public, A. Office, and W. W. W. Afb, "Competing in space," *Ind. High. Educ.*, vol. 2, no. 2, pp. 128–128, 1988, doi: 10.1177/095042228800200219.

[3] G. P. Barnhard and I. Introduction, "toolkit for ISS experiment control," pp. 1–8.

[4] J. P. Trimble and A. K. Henry, "Building a community of open source contributors," *15th Int. Conf. Sp. Oper. 2018*, no. June, pp. 1–14, 2018, doi: 10.2514/6.2018-2508.

[5] D. Laughlin and D. Ph, "NASA GAME CATALOG," no. June, pp. 0–22, 2014.

[6] D. Scarlet, *Chem. Inf. Model.*, vol. 53, no. 9, pp. 1689–1699, 2013, doi: 10.1017/CBO9781107415324.004.

[7] Woolley, B. G., & Peterson, G. L. (2009). Unified behavior framework for reactive robot control. *Journal of Intelligent and Robotic Systems: Theory and Applications*, *55*(2–3), 155–176.

[8] Alzetta, F., & Giorgini, P. (2019). Towards a real-time BDI model for ROS 2. *CEUR Workshop Proceedings*, *2404*(Woa), 1–7.

[9] Martin, J., Casquero, O., Fortes, B., & Marcos, M. (2019). A generic multi-layer architecture based on ros-jade integration for autonomous transport vehicles. *Sensors (Switzerland)*, *19*(1).

[10] "Rocket and Space Technology." http://www.braeunig.us/space/index.htm

[11] "List of space flight simulator games" https://en.wikipedia.org/wiki/List_of_space_flight_simulator_games