International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

The Analysis of the Structure for Testing and Evaluating Multiagent Systems with Autonomous Intelligent Agents

[1]Dr. Mukul Bhatt, [2]Vaishali Singh, [3]Sanjay Bhatnagar, [4]Haripriya V., [5]Zahid Ahmed

Submitted: 10/01/2024 **Revised**: 16/02/2024 **Accepted**: 24/02/2024

Abstract*:* Test and evaluation (T&E) ensure that created systems function as intended in anticipated and unforeseen circumstances. We look at the difficulties in developing a unified framework for testing and evaluating massive cyborg-like physical systems integrated artificial intelligence (AI). We provide a system incorporating testing and analysis into all development and operation phases to help the system learn and adapt in a loud, dynamic, and contested surroundings. The structure conserves testing time and resources when evaluating heterogeneous systems at various hierarchical composition sizes. For a general use case, the framework suggests potential research directions. Statistical modeling, Index systems engineering, AI, experimental design, software engineering (SE), and combinatorial interaction testing. This research proposes a deep learning system for CMF detection that uses CNN and CLAHE to classify photos as genuine or fake. Since some of the hidden elements of the picture are difficult to see using CMF, the CLAHE algorithm brings them to light.

Keywords: *systems engineering, statistical models, SE, artificial intelligence, design of experiments, combinatorial interaction testing.*

1. Introduction

Multiagent systems (MAS) are used in robotics, AI, social networks, and distributed computing. These systems use intelligent, autonomous agents to accomplish objectives. Because of their complexity and significance, these systems need thorough testing and evaluation to guarantee their dependability, efficacy, and performance. This document's structure seeks to fill this requirement by offering a comprehensive framework for testing and assessing MAS [1]. The structural analysis starts with a general comprehension of the multiagent system's layout. The many agents, their functions, and the nature of their interactions with one another and their surroundings must be determined. The architecture includes the agents' decision-making procedures, coordination mechanisms, and communication protocols. This framework helps researchers and developers to acquire useful insights into the strengths and weaknesses of the system by methodically evaluating the behavior and performance of individual agents as well as their collective interactions [2].

In MAS, each "agent" acts and thinks independently. The agent's capabilities, including perception, reasoning, learning, and decision-making, are the primary emphasis of the study. Assessing agent behavior entails looking at how well it fits the system's goals and adjusts to new circumstances. The complexity of agent behaviors and interactions makes testing and evaluating MAS particularly difficult [3].

MAS are autonomous, adaptable, and capable of learning from their mistakes, unlike typical software systems. This calls for a unique strategy that exceeds standard testing methods. The framework evaluates MAS theoretically and practically [4]. It encompasses agent behaviors, communication protocols, coordination, decision-making, and system performance. These criteria are used to assess the system's capabilities and applicability comprehensively. MAS are tested in real-world simulations and scenarios. It acknowledges that such methods perform best in dynamic, unpredictable environments [5]. As a result, the framework suggests using test beds and simulation platforms that accurately represent the challenges and complexity of the target area.

2. Related work

The research [6] provided deals with the social level as an organizational model and considered a particular method for evaluating a dimension. Any MAS has three basic dimensions at a minimum: the individual and social classes and communication interfaces. The Organization-based Multiagent Software Engineering (O-MaSE) methodology framework, which incorporates numerous useful technologies to encourage industry adoption, is suggested in the article [7]. O-MaSE is an extensible agent-oriented

[1]*Department of ISME, ATLAS SkillTech University, Mumbai, Maharashtra, India, Email Id- mukul.bhatt@atlasuniversity.edu.in, Orcid Id- 0000-0002-5511-7551*
[2]*Maharishi University of Information Technology, Lucknow, India, Email Id- singh.vaishali05@gmail.com, Orcid Id- 0000-0001-8304-8947*
[3]*Chitkara University, Rajpura, Punjab, India, sanjay.bhatnagar.orp@chitkara.edu.in, https://orcid.org/0009-0004-7474-1511*
[4]*Jain (Deemed to be University), Bangalore, Karnataka, India, Email Id- v.haripriya@jainuniversity.ac.in, Orcid Id- 0000-0003-2035-2452*
[5]*Vivekananda Global University, Jaipur zahid.ahmed@vgu.ac.in, Orcid Id- 0009-0002-4020-1002*

approach built on dependable, well-defined ideas supported by agent tool plug-ins. The research [8] proposed two various multiagent architectures for adaptive intelligent signal management that have been applied to a simulated complicated urban traffic network in Singapore explored. The results show that the multiagent signal controller performs better than pre-timed and signal control techniques. The article [9] determined an assessment framework and systematic procedure for evaluating already-defined or newly-defined domain-specific modeling languages (DSML) for MASs. Both the language and the related tools are considered specifically for MAS DSMLs. The research [10] presented a computational design method for conceptual building design. Several environmental performance indicators are used in a multiagent system design toolkit to create design ideas and assess them. The research [11] examined Analytical Hierarchy Process (AHP) based comparative MAS DSML assessment technique. MAS DSML multi-criteria decision-making criteria are categorized. The method lets developers prioritize these factors according to their modeling language requirements and evaluate DSML alternatives. Automatic significance distribution computation determines the best DSML.

The article [12] proposed how PADE-based network service agents were developed and used. A project's GitHub URL was also provided. Algorithms for deep learning and machine learning are evaluated in the implementation. Many sophisticated algorithm research may be conducted in this context. Tools and libraries for developing and testing MAS are available via the Java-based Multiagent Development Kit (MADKIT) framework. In addition to simulation, monitoring, and debugging tools, it provides various features for creating and implementing intelligent agents [13]. The Java-based MAS framework is Java Agent Development Framework (JADE). It provides resources for coordination, mobility, and agent communication. To test and evaluate agent-based systems, JADE simulates them [14]. Agent-based simulation toolkit called Recursive Porous Agent Simulation Toolkit (REPAST). REPAST is well-liked.

Design and testing MAS enables researchers to define agent behavior, interactions, and parameters. Simulations, data gathering, and analysis are all possible using REPAST [15]. The research [16] presented here suggests a MAS for intrusion detection that combines the strengths of MAS with the precision of deep learning. As a result, we created autonomous, intelligent, and adaptive agents by combining the k-nearest neighbor, multilayer perceptron, and autoencoder algorithms. To properly recognize and classify complicated network attacks, the research [17] offers a deep reinforcement learning-based Intrusion Detection System (IDS) that uses Deep Q-Network logic in many distributed agents and attention approaches. To provide scalable, fault-tolerant, multi-view architecture-guided security, our multiagent IDS is a distributed threat detection platform with agents working together. The research [18] provided that healthcare is a topic of discussion at international conferences, in academic publications, and the real world. Five subdomains and three systems were assessed. They are utilizing similar parameters to compare these systems. MAS for healthcare is suggested. The article [19] adaptable multiagent networks for smart cities based on existing literature. In the related literature, designing and managing adaptive systems is classed. These procedures outline, keep track of, organize, and evaluate the operation of autonomous MAS. The article [20] provided the mature and strong commercial product Jack Audio Connection Kit (JACK) agent platform. JACK addresses industrial adoption needs, including familiarity, scalability, and integrability. They also discuss JACK's structural capabilities and JACK's hierarchical teamwork.

3. Descriptive use case

Our model is based on the typical example of a satellite network's usage with a variety of autonomous intelligent agents (AIA) working independently and reporting to a central controller. Every satellite has sensors, actuators, and software at the local hierarchical scale, combining predictive management and artificial intelligence anticipated to adapt following installation owing to shifting surroundings and knowledge accumulation (shown in fig.1). The components of each subsystem may be divided into hardware and software functions. The kinds, numbers, and locations of each satellite and other status data, like whether the satellite has been harmed or its software corrupted, may all be used to characterize a system at the global hierarchical level. Observing aims, enemy powers, and visibility are possible operational environmental elements.

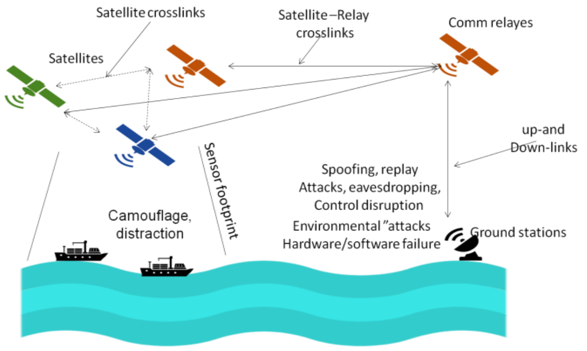
Satellite crosslinks
Satellite –Relay crosslinks
Comm relayes
Satellites
up-and Down-links
Sensor footprint
Spoofing, replay Attacks, eavesdropping, Control disruption
Camouflage, distraction
Environmental "attacks Hardware/software failure
Ground stations

Fig.1. Multi-agent satellites system use scenario illustration

4. Video Transmission Protocol Model

T&E must be included during the system's development and life cycle. Given that it is a well-known systems engineering model, the "Vee" model of systems engineering provides a strong base for building a unified framework. The hierarchy's system levels are connected to several groups of validation and verification, by the "Vee" paradigm.

Instead of waiting until the full system is finished, specifications and test schedules are generated early in development. For instance, tests that check system performance are created at the beginning of the project when needs for the complete system are determined, yet they are not conducted until close to the finish. As a result, Moving along the left side of the "Vee," the system is planned from the top down as tests are simultaneously being generated for that slice. Although the subsystems are constructed, and the system is linked, test execution is done "bottom up," moving along the first row on the "Vee." Testing multiagent AIA systems fails due to the "Vee" model's hierarchical decomposition assumptions. Most systems development life cycles involve concept research, technological development, conceptual design, detailed design, manufacturing, packaging, testing, launch, operations and sustainment, and closeout. Similar to the SE Waterfall Model, identification of requirements, design, implementation, testing, and maintenance are all parts of the development process. These methods assume that all assessment criteria might be provided during a requirements phase and decomposed into separate either parts or subsystems. Maintaining a thing enables "fixing" it. Before deployment but ignores changes in agent behavior after distribution. AIA systems with several agents defy these presumptions. It could be hard to decide how the system should adapt since the environment changes. The threat capabilities of advanced foes could be unpredictable. Combinations of subsystems satisfy needs. Satellite ensembles can complete a task because of their varied positions and qualifications. Following the satellite after the program is deployed, it perhaps altered by earth station software pushes or by the embedded AI interacting and learning from other satellites in the constellation. Systems that are difficult to recall do not lend themselves to incremental or iterative techniques. Even though subsystems like satellite parts or software are planned, developed, and extensively tested in real-world scenarios, the system is only deployed when the satellite is launched. As a result, a crucial element of our technique up to deployment is the system engineer "Vee" model, with some iteration. We provide a "T" phase to test for events during operation. The running of all relevant tests from the last step and additional tests to accommodate the code changes may be necessary to achieve an objective mission amendment. Testing is required to ensure that embedded AI software picks up the "right" behaviors. Regular testing is essential for fault discovery and mitigation when debris and hardware degradation happen. The last possible assault is on communication systems, necessitating new intrusion detection method signatures. Analyze embedded AI's resilience to unknown vulnerabilities or development issues like data drift. Feedback ought to come next. They are learning from existing systems loops to deployment at the "P" stage. This loop's extension may serve as a development roadmap. The framework's VTP model is seen in Fig.2.

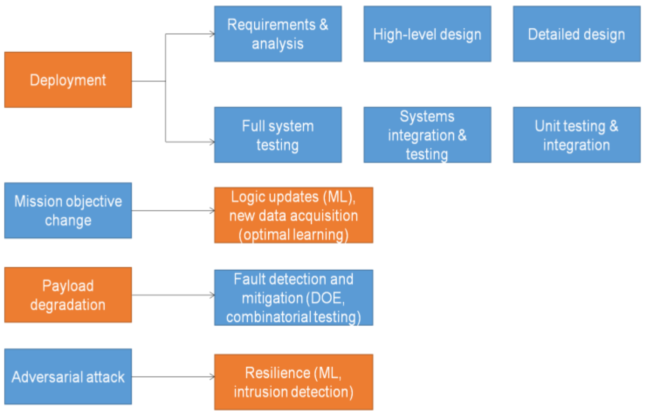
Deployment
Requirements & analysis
High-level design
Detailed design
Full system testing
Systems integration & testing
Unit testing & integration
Mission objective change
Logic updates (ML), new data acquisition (optimal learning)
Payload degradation
Fault detection and mitigation (DOE, combinatorial testing)
Adversarial attack
Resilience (ML, intrusion detection)

Fig.2. The VTP framework extends the “Vee” model to include testing throughout system deployment and a feedback loop

5. Specialization of Study

Cyber-physical systems incorporating AI are known as AIAs. The testing and evaluation of the integrated system should use techniques of testing that are scientifically valid for each component and consider the testing needs of its mechanical, deterministic, AI, and electrical components. T&E should be used to assess algorithmic flaws for embedded AI. When using neural networks for learning, an assessment technique is required to gauge performance sensitivity to input changes or noise. This is crucial to detecting data contamination and calculating mission success under tense circumstances. Considering sensor input and controlling actuators to enable the agent to interact with its surroundings and exchange information with other agents in the MAS, other deterministic software supports the AI capabilities of the software. Coding may be done using white box methods focusing on structural code coverage. For COTS or vendor-supplied software, it is necessary to utilize equivalence partitioning, state transition testing, use case testing, boundary value analysis, decision tables, and combinatorial methods. The variation of the response variables is assigned to independent variables, or factors, through statistical analysis and assesses system performance based on factor levels when analyzing physical components. Design of experiments (DOE) is an analytical framework for evaluating test adequacy. It is a systematic process for choosing test cases that span the functional area. DOE uses embedded software to evaluate complex systems. Tests to learn about the system are the first step toward optimal learning. After learning a Bayesian substitute for the goal function, the following tests are selected using a heuristic that iterates over the test space with the greatest uncertainty and uses regions that maximise the goal-oriented function. The plans must also look at the autonomy of the whole integrated system of systems. Although each subsystem may be isolated in its field and use approved testing techniques, the hybrid system crosses several fields, therefore T&E must take it into account. the whole thing. Testing may be necessary depending on how various systems interact. For instance, a satellite may need to comprehend that altering its position can help it to get around sensor visibility restrictions. Coordination comprising motors, control software, integrated AI, and sensors is required to validate this behavior. The list still needs to be completed. AIA usage situations can need additional fields. To develop tests that address particular difficulties, such as attention problems and how to use the human-computer interaction people and computers may communicate and grasp collaborative objectives, researchers evaluating AIAs with significant human-in-the-loop elements or with people should consider psychological testing literature. Psychological Subject Matter Expert (SME) consulting may be able to define criteria for AIA learning and ensembled AIA collaborative behaviors even in the absence of human involvement.

6. Chart of test

Testing is carried out at every stage of development to identify and eliminate issues quickly. The smallest testable components are subjected to unit testing using simulated inputs or digital surroundings. When integrated into subsystems, parts should function as planned. However, interactions may result in failures or performance problems. For instance, COTS control software could not assume the input range of a satellite sensor. When combined, the code can crash or act strangely. Test suites are available using combinatorial interaction testing (CIT) to methodically identify issues brought on by combining several interacting parts up to a certain interaction size. To create test suites using tools like Automated Combinatorial Testing for Software, testers must first define the components, levels where they should be evaluated, the maximum interaction size, and limitations. The tester must be able to discretize continuous levels and know the elements or aspects of interest in CIT. A system becomes a component in a complex system via integration testing. AIA satellites come equipped with control software, actuators, sensors, and actuator actuators at the local level. System performance is ensured via integration verification and validation testing. After being deployed into a constellation, each satellite transforms into a component of the global system.

Interaction testing determines if the constellation's sensor coverage is enough for a tactical operation or whether satellite-ground station communication relays function. Global environmental factors, such as sight-impairing storms, may compromise the effectiveness of a mission. Adversarial attacks could be included in simulations. After deployment, interactions between subsystems directly below and external variables like storms that impact the failures at the global level of the hierarchy may be brought on by particular satellites' sensor visibility or ground station connectivity. Approaches for CIT fault localization may identify the interactions causing the issue. An error may move up the hierarchy from a component system lower down. It could be necessary to descend through the interactions of the order to locate the defective part.

7. Success of the test plan

Rigorous testing in a variety of situations ensures system performance. The system and relevant ambient elements define the test input space and their levels. As a result, the multidimensional test input space is covered by test plan points. Most tests are costly, time-consuming, and resource-intensive. An exhaustive investigation of the impact of variables and factors at the variable and factor level on system performance is possible thanks to a DOE full factorial design. Comprehensive testing is not achievable for complex systems with many groups and components. In fractional factorial designs, certain effects are aliased, and variation cannot be entirely allocated. To provide appropriate knowing the system with fewer test points, the percentage may produce anomalies between impacts of more complex interactions that are not anticipated to be substantial and big impacts. Reduces uncertainty and determining the ideal system performance factor values are the goals of optimum learning. With fewer experiments, the exploration-exploitation heuristic technique may locate the best location for a component. Though it draws, CIT often uses a covering array, a multifunctional array in which the levels specified for the factor under consideration in that test are reflected in each cell's values, with the columns denoting variables and the rows denoting tests. This is a combinatorial array that represents factors, tests, and levels. Since each possible set of values for interacting characteristics up to a specific strength is tested, conclusions about performance in the remaining working area from fewer tests using physical principles and prior knowledge. An encompassing array, a multifunctional array in which the numbers in each cell represent the level provided for the factor in question in that test, and the rows and columns imply factors and challenges, respectively, is a common tool CIT uses when designing test suites. Considering that any set of values for interacting features to a certain level occurs in at least one test to detect failures produced on by interactions up to the required strength, a covering array may be utilised. One row generates a test if there are more variables than there are variables with value. suite containing some of the factorial's tests and covers some interactions. Trials expand logarithmically when factors are added. Interactions coverage is provided by a covering array, which serves as a test suite. However, defect discovery is not guaranteed.

Humans often participate in intricate systems. To avoid aliasing with ambient variables and levels, between-subject and within-subject experimental design approaches describe and separate human contributions to test findings. While within-subject designs pair persons with factors and levels for comparison, between-subject structures assign individuals randomly to certain combinations of variables and classes. The advantages of each approach above are combined in our framework to integrate data choose the appropriate test length at all test slices are different growth phases, and cover the input area systematically at both the global and local levels. To avoid extensive testing later on lower-hierarchy components, find errors early in development. At lower levels of the hierarchy, as inside a sensor function, the test input area is condensed. Tests may be accelerated and automated using scripts or simulated inputs, and test psychics can be calculated. Component interactions are the main focus of subsystem testing. Focusing on the most crucial elements may be necessary when the global system is tested for mission accomplishment. These data support extensive component testing but less rigorous global testing. The finest DOE designs for lower-hierarchy components are full factorial or fractional factorial designs. Using CIT techniques, integration interactions could restrict the exponential expansion of test points. Global upon system installation, system behaviour could not be known dynamic environments with learning AI components. When resources permit, optimal learning may estimate system limitations and adaptively choose tests. The framework strikes a compromise between test point count and system integrity, as shown in Fig.3. While testing a fully realized system is more difficult and contains fewer test points, the results are more accurate.

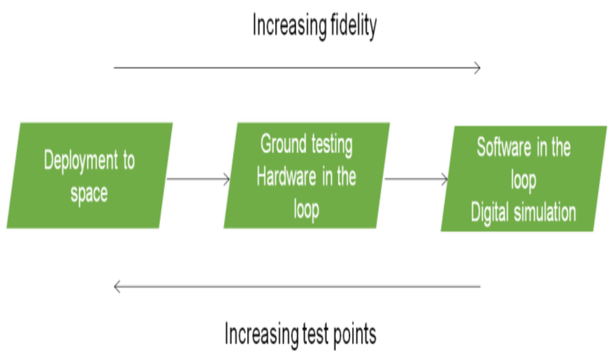
Increasing fidelity
Deployment to space
Ground testing Hardware in the loop
Software in the loop Digital simulation
Increasing test points

Fig.3. Reliability replaces test points when system design hierarchy rises

8. Design Framework for Tests

There are no test lists in the framework. Instead, the framework lists important considerations for thorough test plan creation. These considerations for the VTP model slice will lead to various test designs. Future tests must be designed using operational test slice data during the "P" phase of the VTP model. To identify adversary compromise, a system may use testing for combinations of interactions and programmed checks of each agent in its deployed form. These tests will guide algorithm changes and prompt another round of objective testing to confirm agent upgrades.

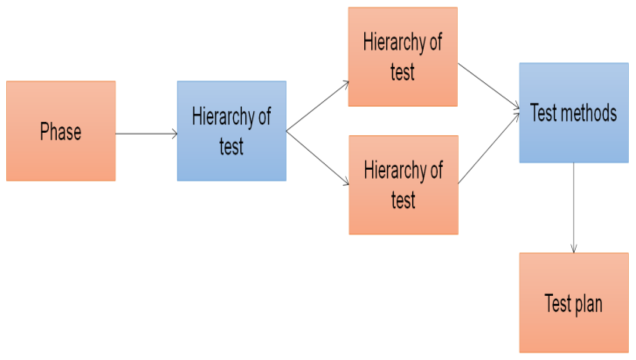

Fig.4. Framework that governs test planning.

The framework creates a test plan, as shown in Fig.4. The domain of study and test hierarchy is based on the current life cycle phase of the system under test (SUT). Testing tiers and SUT component identification is determined by the test hierarchy. The input space for the test consists of elements, levels, and connections. Considering the expense of doing every test at the present size and assuming that lower-hierarchy systems operate as anticipated, test plan efficiency dictates the test. The structured approach establishes appropriate test methodologies and goals. The SUT research topic and test hierarchy determine the test objective.[21]Goals and test plan effectiveness determine test techniques. All available historical data is used to develop the test plan. Three sample situations from the AIAs life cycle's MAS are shown in Table.1.

Table.1. High-level explanation of how each framework idea helps to direct the creation of test plans

System under test	Phase	Field of study	Hierarchy of test	Test plan efficiency	Goal of test	Test methods
Ground station operators for Constellation	Operation	Psychology (AIA teaming, HCI)	Satellites, ground stations, operators, environmental conditions	Operational system Focus on mission goals, performance modifications	Mission success	Human-factor DOE (subject design)
Al algorithm	Unit testing	AI mission knowledge	Code functions are components. Software integration Functions artificial inputs	Black box testing using scripts with fictitious programme inputs To highlight the variety of inputs and situations conceivable, concentrate on flaws or poorly understood needs.	Analyse the biases in AI and its robustness, performance, and dependability.	CIT paired with ideal learning to maximise performance and minimise uncertain performance areas
Satellite	Testing of Systems Integration	Above plus: • Electronic hardware • Software • Mechanical systems	AI programme, sensors, motors, and control software are the parts. Integration takes into account how parts send and receive information from other elements.	Full system integration tests in a virtual space setting Pay attention to how systems connect with each other to find problems or drops in performance.	Check the system's speed, dependability, security, and ability to work with other systems.	DOE mixes well-known software interaction areas with factors about the surroundings.

The list of academic disciplines that make up the SUT and from which testing methods should be taken needs to be completed. Remember our use case, a satellite network with point detection and wide-area search. The satellite network intended to monitor The network performs wide-area searches to comprehend typical traffic patterns and find actions that could point to possible illicit shipping operations. After spotting an anomaly, the network has two major goals: to keep searching a wide region and to keep tabs on the suspicious vessel. We can navigate with this background and suggest logical test design architectures. Early testing of AI algorithms is seen in the first row. The central controller allocates satellites in our use case to provide widespread coverage and tracking. We investigate the main controller's performance, biases, resilience, and reliability. We can afford a method that explores high-uncertainty areas with optimal learning and covers all bases with CIT since simulations are inexpensive. The CIT architecture may use CIT at a high intensity (surrounding several contacts) to integrate unusual activity in the shipping statistics from the past. The nation, kind, size, and geographical position of the vessel may all be included in the CIT test set. The CIT is augmented by optimum learning in industries with significant uncertainty or rapidly changing AI algorithm performance. How issues are impacted by satellite-scale testing is seen in the second row of Table.1. It is now necessary to test algorithm performance and system integration. New standards and test formats are required. The system-centric test design uses an excerpt of the CIT AI testing scenarios. When simulating the space deployment, a hardware-in-the-loop test facility may mimic the inputs of satellite sensors. Active adversary, weather influences on inputs, AI performance scenario, etc. Experimental designs look at how the system performs while running AI algorithms with simulated inputs. We need to comprehend the connected system's mission accomplishment after deployment. The need for ground control station human integration for mission success. The availability of skilled constellation mission controllers limits our test size. We assign ground teams utilizing a within-subject strategy to deployed system situations. The constellation performs control functions while in operation. The three scenarios show how to create a test design strategy for a complex system-of-systems at various sizes that adheres to Utilising the test design framework and the slice in the VTP model, the test objective, which is generated from the field of research and the hierarchy of tests, with the expected test efficiency. Future scope

Threshold designing, developing, and deploying the VTP model, this framework considers the research field, the hierarchy of tests, and the effectiveness of the test plan to provide an entire test plan. This method combines T&E methodology and applies to many multiagent AIA scenarios. In a series of studies, we showed how the framework might be used for a group of satellites that performs point detection and wide-area search. To determine if the framework can be used for complicated systems and to determine whether any features are required in order to guide the construction of test plans, additional use cases must be tested against the framework. Problems peculiar to the setting gave rise to two new study areas. It may be possible to do "symmetrical" tests that effectively repeat the same configuration when integrating a system of systems with comparable subsystems. Consider testing the integration of two satellites, S_1 and S_2 with relative component configurations but at different orbital positions, P_X and P_Y. They share actuators, control software, algorithms, and payload. Without any restrictions, tests should be generated that include both combinations of the satellites. Fig 5. and Table 2 Shows the impact factors of autonomous intelligent agents refer to the criteria or metrics used to assess and evaluate the influence, effectiveness, or significance of these agents and their actions in achieving desired outcomes. Aspects affecting the influence of self-sufficient intelligent agents refer to the factors that can impact the degree of influence these agents have in their decision-making processes, task execution, and overall impact.

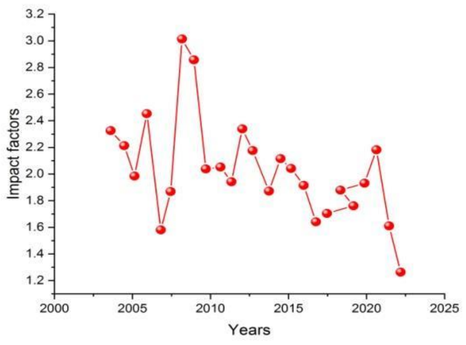

Fig.5. Impact factors of autonomous intelligent agents

Table.2. Aspects affecting the influence of self-sufficient intelligent agents

years	Impact factors
2005	1.95
2010	2.05
2015	2.10
2020	1.76

An autonomous intelligent agent's impact factor is any measure or standard by which the agent's influence, efficacy, or relevance in producing an intended result may be determined and evaluated in fig 6. Quantiles, in the context of self-reliant cognitive agents, are statistical measures used to divide a distribution of performance or capability into equal intervals shows in table3.

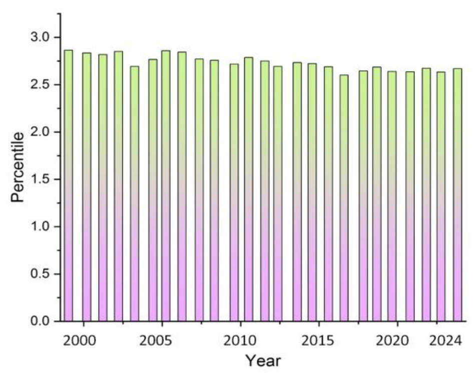

Fig.6. Percentile for autonomous intelligent agents

Table 3. Percentile for autonomous intelligent agents

years	percentiles
2005	2.8

2010	2.7
2015	2.6
2020	2.7

9. Conclusions

Both satellites are replaceable for testing purposes early before learning or damage. So neither observation is required. Just the relevant instances should be observed and examined when testing is expensive. So, a tool for creating test suites is necessary to prevent symmetrical tests. Using constraints to solve this problem and create a test suite may be computationally expensive in large constellations. We postulate that a partial ordering on certain components could result in a sequence covering array-inspired covering arrays devoid of symmetrical checks. To assess how much of the variation, we suggest analyzing the framework in the context of the satellite use case to see how well it captures success and failure and whether task-scale testing can be used to predict mission success at the top of the hierarchy. They limit most system modifications because large-scale testing may be costly and only practical after the constellation is put into space.

References

[1] Rocha, J., Boavida-Portugal, I. and Gomes, E., 2017. Introductory chapter: Multiagent systems. In *Multiagent Systems*. IntechOpen.

[2] Alaca, O.F., Tezel, B.T., Challenger, M., Goulão, M., Amaral, V. and Kardas, G., 2021. AgentDSM-Eval: A framework for evaluating domain-specific modeling languages for multiagent systems. *Computer Standards & Interfaces*, *76*, p.103513.

[3] Qasem, M.H., Hudaib, A., Obeid, N., Almaiah, M.A., Almomani, O. and Al-Khasawneh, A., 2022. Multiagent systems for distributed data mining techniques: an overview. *Big Data Intelligence for Smart Applications*, pp.57-92.

[4] Calegari, R., Ciatto, G., Mascardi, V. and Omicini, A., 2021. Logic-based technologies for multiagent systems: a systematic literature review. Autonomous Agents and Multiagent Systems, 35(1), p.1.

[5] Esposito, D., Schaumann, D., Camarda, D. and Kalay, Y.E., 2020. Decision support systems based on multiagent simulation for spatial design and management of a built environment: the case study of hospitals. In Computational Science and Its Applications–ICCSA 2020: 20th International Conference, Cagliari, Italy, July 1–4, 2020, Proceedings, Part III 20 (pp. 340-351). Springer International Publishing.

[6] Gonçalves, E.M.N., Machado, R.A., Rodrigues, B.C. and Adamatti, D., 2022. CPN4M: Testing Multiagent Systems under Organizational Model M oise+ Using Colored Petri Nets. Applied Sciences, 12(12), p.5857.

[7] DeLoach, S.A. and Garcia-Ojeda, J.C., 2010. O-MaSE: a customizable approach to designing and building complex, adaptive multiagent systems. International Journal of Agent-Oriented Software Engineering, 4(3), pp.244-280.

[8] Balaji, P.G. and Srinivasan, D., 2010. Multiagent system in urban traffic signal control. IEEE Computational Intelligence Magazine, 5(4), pp.43-51.

[9] Challenger, M., Kardas, G. and Tekinerdogan, B., 2016. A systematic approach to evaluating domain-specific modeling language environments for multiagent systems. Software Quality Journal, 24, pp.755-795.

[10] Pantazis, E. and Gerber, D., 2018. A framework for generating and evaluating façade designs using a multi-agent system approach. International Journal of Architectural Computing, 16(4), pp.248-270.

[11] Asici, T.Z., Tezel, B.T. and Kardas, G., 2021. Using the analytic hierarchy process in evaluating domain-specific modeling languages for multiagent systems. Journal of Computer Languages, 62, p.101020.

[12] Hettige, B., Karunananda, A.S. and Rzevski, G., 2021. MaSMT4: The AGR Organizational Model-Based Multiagent System Development Framework for Machine Translation. In Inventive Computation and Information Technologies: Proceedings of ICICIT 2020 (pp. 691-702). Springer Singapore.

[13] Antelmi, A., Cordasco, G., D'Ambrosio, G., De Vinco, D. and Spagnuolo, C., 2022. Experimenting with Agent-Based Model Simulation Tools. *Applied Sciences*, *13*(1), p.13.

[14] Bellifemine, F., Poggi, A. and Rimassa, G., 2001. Developing multiagent systems with JADE. In Intelligent Agents VII Agent Theories Architectures and Languages: 7th International Workshop, ATAL 2000 Boston, MA, USA, July 7–9, 2000 Proceedings 7 (pp. 89-103). Springer Berlin Heidelberg.

[15] Simmonds, J., Gómez, J.A. and Ledezma, A., 2020. A brief review of the role of agent-based modeling and multiagent systems in flood-based hydrological problems. Journal of Water and Climate Change, 11(4), pp.1580-1602.

[16] Louati, F. and Ktata, F.B., 2020. A deep learning-based multiagent system for intrusion detection. SN Applied Sciences, 2(4), p.675.

[17] Sethi, K., Madhav, Y.V., Kumar, R. and Bera, P., 2021. Attention-based multiagent intrusion detection systems using reinforcement learning. Journal of Information Security and Applications, 61, p.102923.

[18] Jaleel¹, H.Q., Stephan, J.J. and Naji, SA, 2023, June. 1 Baghdad College of Medical Sciences, Baghdad, Iraq 2 Al-Esraa University College, Baghdad, Iraq 3. In *New Trends in Information and Communications Technology Applications: 6th International Conference, NTICT 2022, Baghdad, Iraq, November 16–17, 2022, Proceedings* (p. 107). Springer Nature.

[19] Nezamoddini, N. and Gholami, A., 2022. A survey of adaptive multiagent networks and their applications in smart cities. Smart Cities, 5(1), pp.318-347.

[20] Graham, J.R., Decker, K.S. and Mersic, M., 2003. Decaf-a flexible multiagent system architecture. Autonomous Agents and Multiagent Systems, 7, pp.7-27.

[21] Dr. Vishnu Rajan., &amp; Dr. Gokula Krishnan, V. (2022). A study on the use of machine learning and complex hierarchical structures to visualise text categorization. Technoarete Transactions on Advances in Computer Applications (TTACA), 1 (2), 8-16.