Int j simul model 21 (2022) 1, 41-52
Original scientific paper

COUPLING OF ODE AND DES MODELS FOR SIMULATION OF AIR DEFENCE IN WAR-GAMING EXPERIMENT

Stefek, A.[*]; Casar, J.[**]; Stary, V.[**] & Gacho, L.[**]
[*] Department of Informatics and Cyber Operations; University of Defence, Kounicova 65, 662 10 Brno, Czech Republic
[**] Department of Air Defence; University of Defence, Kounicova 65, 662 10 Brno, Czech Republic
E-Mail: alexandr.stefek@unob.cz, josef.casar@unob.cz, vadim.stary@unob.cz, lukas.gacho@unob.cz

Abstract
The planning stage of any procedure is critical to its successful and efficient execution. Even in the military, over the last few decades, mission planning support has evolved rapidly using modern Modelling and Simulation (M&S) tools. The article focuses on the M&S of Surface Based Air Defence (SBAD) and the design of the complex hybrid simulator coupling Ordinary Differential Equation (ODE) and Discrete-Event Simulation (DES) models. The simulator includes all entities and procedures that participate in SBAD missions, specifically the ODE models of a target, missile, radar, and the models of Command and Control (C2) system procedures. The design of the simulator also includes the possibilities of automated generation of flight routes, fire control and target distribution for multiple targets and Fire Units (FU) using queuing theory algorithms. As a main platform for the development and scenario design, the Jupyter notebook environment with the Python programming language was used. The primary objectives are to design, validate, and implement the proposed simulator in SBAD mission planning, military cadets' education, and experimental tasks.
(Received in October 2021, accepted in January 2022. This paper was with the authors 1 week for 1 revision.)

Key Words: Modelling and Simulation, Flight Route, War-Gaming, Optimal Track, Air Defence, Command and Control

1. INTRODUCTION

War-gaming in connection with M&S is nowadays inevitable part of any modern military mission analysis or planning, during its Course of Action (COA) procedures [1]. In some aspects, the war-gaming is very close to civilian management and planning, e.g. logistics [2], job production [3] and uses established methods and algorithms such as route planning optimization [4].

There are several specialized M&S instruments on the market, which are used in security and military domain, e.g. tactical constructive simulators, such as Virtual Battle Space (VBS), Presagis STAGE or constructive war-gaming platform MASA SWORD. These platforms are very robust, complex and they are used in various applications, including war-gaming and COA planning. This robustness and complexity are linked with the higher user, technical and time requirements and usage, which is more difficult. These are the main disadvantages of mentioned types of Commercial Off-The-Shelf (COTS) solutions. The higher prices and unknown inner structure of these black box solutions are another negative attributes. In addition, such a software system due to its demands usually disallows its integration into systems for commander decision support.

The area of SBAD is unique due to its complexity, which is based on three essential pillars: the sensors, effectors and C2 systems. These pillars are affected not only by their technical aspects but considerably also by the human factor. The M&S of SBAD and its pillars is fundamental qualification for further optimisation process of complex structures and connections of sensors, effectors and C2 systems. Due to this complexity, different parts of SBAD systems are modelled and simulated with various techniques. The sensor part is more suitable for DES models and the effector part is applicable for ODE models. The C2 procedures

and connection between models using addition methods and algorithms, which are closely described in Chapter 2.

1.1 Main aim of the research

Presented article deals with the development of Experimental/Educational Air Defence Simulator (EXEDAD SIM) using results of several previous researches about C2 structures [5], C2 simulations [6] and flight route planning [7]. The simulator allows effective and simple usage based on open platform and it can be easily modified according to the assigned task. Additional benefits are also its usage for educational purposes and possible application for autonomous processing or Artificial Intelligence (AI) algorithms training [8].

Even though the human experience is the most valuable element in decision making process, the basis data with suggested solutions can be produced automatically, and the commander will make the final decision.

1.2 Concept of the EXEDAD simulator

The EXEDAD SIM is designed as a hybrid simulator with three main parts. The first part is the Event Calendar, which creates and arrays the events of entities in chronological order in the simulation. The second part is represented by ODE models of entities (Missile [9], Aircraft [7], Radar [10]) with its solvers, track and destination controllers. The third part perform as a Command Post (CP) / Fire Distribution Centre (FDC), where the C2 procedures are included, there is also applied the queuing theory and the probability distribution.

Despite the fact that the SBAD operations are very diverse and affected by number of internal and external factors, they can be simplified into several steps with following entities, see Fig. 1 where RADAR, CP/FDC and FU are represented by NATO Tactical Symbology [11].

There are several elemental steps in the SBAD operation procedures. The first one is to detect an Aerial Object (AO) by any type of sensor (radar, visual, IR, etc.). The next step is to identify and classify the object as friendly or hostile. This is provided at CP by the Identification Friend or Foe (IFF) [12] system or by additional procedures (flight plan comparison, visual identification, etc.). In the event that the AO is identified as hostile, the following action has to be taken. The target is assigned to the effectors (FU) in accordance with defined rules of engagement, and in the case of possible engagement by the FU, the target is engaged. If the engagement is not possible, the target is passed to another FU, or to the queue, and it is processed according to the Queuing Theory (QT) [13].

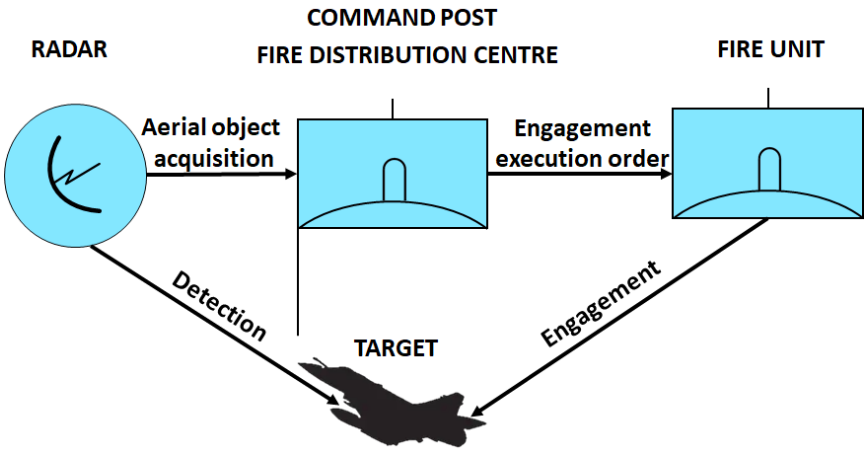

Figure 1: Simulator entities and functionalities.

1.3 Modelling and simulation tools

The Python programming language was chosen as a primary tool for the effective M&S and a simulator development. The Python language is one of the most growing programming language. It is designed, developed and supported by well-structured community and becomes important in a field of data science.

The Python language was extended to support remote computation (the remote kernels concept) accessible within web interface. This environment named Jupyter is actually wide spread among many universities and companies. As an example a project in Canada [14] or the Colab project from Google [15] can be mentioned.

It is becoming common to attach a set of well-documented experiments ready to rerun (reproducible science) to a published paper. As an example the cooperation between IEEE and CodeOcean could be mentioned [16].

The Jupyter environment offers number of benefits like language simplicity, robustness, world-wide usage, user support and allows the creation and sharing of interactive documents with live code, equations and visualisations and the output is formatted as an interactive document for further applications e.g. educational [17, 18].

The text part of the article did not include the simulator code. However, it includes the description of algorithms, models and the simulator code is available as a standalone Python library named "simodes" [19].

2. EXEDAD SIM LAYOUT

As it was noticed in the introduction, the EXEDAD SIM is designed as a hybrid simulation. Fig. 2 represents the overall structure of EXEDAD SIM and it is based on idea published in [7]. The bold full line blocks represent the individual entities (Target, FU, RADAR, CP/FDC), the thin full line blocks are the main models and simulation parts (Path Controller, Destination Controller, Generated Path, Event Calendar, etc.). The dashed line block expresses the entire EXEDAD SIM with possible AI/Autonomous evaluation.

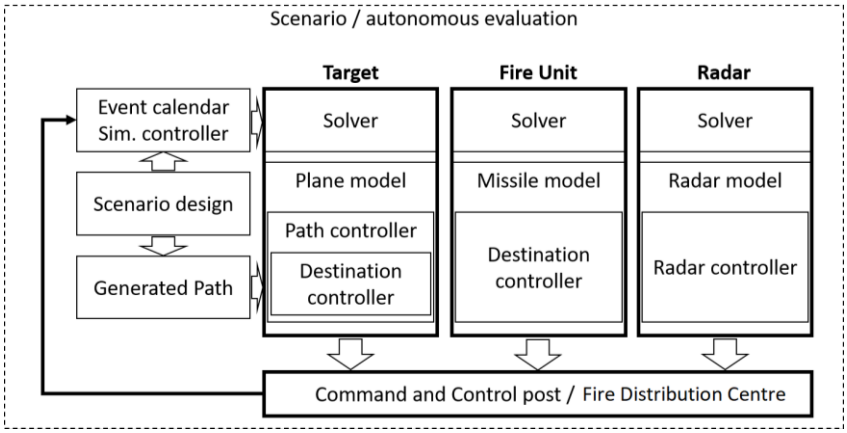

Figure 2: EXEDAD SIM general schema.

2.1 Event Calendar

The DES forms the core of the simulator, called the Event Calendar, and it is composed of all simulation events. These events (entities) are served at the corresponding time in chronological order. Managing (synchronising) several dynamic systems (described by ODE) as a complex structure with several timelines or subsystems with DES models proceeds to a complex solution which is described as an event manager (part of an event calendar) and it is used during event-driven simulation. An ODE model with its solver is implemented as an infinite generator,

periodically releasing a new state of the dynamic system and the time at which this state will be valid. In fact, this ODE model with solver conversion into a generator creates a source of timeline, which the Event Calendar must synchronize with others.

The class Event Calendar [19] contains subsequent methods:

- Method Step – classifies the queue by time, removes the last executed event, and calls the current event.
- Method PrepareNext – prepares the next event and returns its initial time.
- Method AddEvent – adds an event to the event queue with the time of its execution.

The Event Calendar establishes the planning and execution of events and calls the dynamic models from the continuous simulation part.

2.2 Flight simulation

Flight is the singular component that defines entity movement. The continuous part contains the dynamic model of a plane with its controller (defined by its ODE) and the solver, which calculates the trajectory based on the user-made scenario and Way-Points (WP) in the Path Generator, see Fig. 3 and [20].

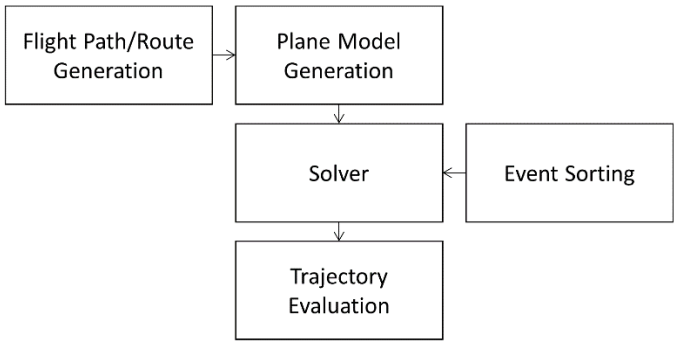

Figure 3: Plane flight simulation functionality schema.

The model of a plane is presented as a mass point with five Degrees of Freedom (DoF) based on the Pitch/Yaw kinematic model, where the roll attribute is not used. The generic five-DOF model (1) was chosen to keep the simulation as simple as possible for verification. This model can be easily replaced with a six-DOF Pitch/Yaw/Roll model or a different one if needed. The model consists of a pure plane model and a path and destination controller. The model fits the standard mathematical ODE structure, Eq. (2), thus, it returns state derivation, Eqs. (3) to (5), at a given time and state. The controller output values are yaw omega and pitch omega. The values (state variables) of roll, omega, and velocity were omitted, as they were found avoidable for this experiment as well as for the XYZ model.

2.3 Plane model

The plane model controller ensures following the path using high-order function (plane model). The controller takes a function as a parameter (variable) and returns a function using Python yield statement. The resultant Generic Plane Model (GPM) is defined by a differential equation with the following nine variables (*state*):

$$state = [x, y, z, vx, vy, vz, v, pitch, yaw] \tag{1}$$

where:

- x, y, z are the plane position in Cartesian coordinates [m],
- v_x ,v_y, v_z and v (meaning vector v) represents the velocity [m/s],
- *pitch* stands for a plane elevation (nose up or tail up) angle [rad],
- *yaw* is a plane azimuth (side to side) angle [rad].

The parameters one to six (x, y, z, v_x, v_y, v_z) depends on the core parameters seven to nine (v, *pitch*, *yaw*) and vice versa. The plane model according to the plane type also defines basic limitation factors and initial conditions. For the purpose of the simulation verification the minimum velocity limit is *EpsVmin* = 10 m/s and maximum centripetal acceleration of yaw and pitch is *AYawmax* = *APitchmax* = 20 rad/s^2.

The ODE model (state function) is represented by following equations:

$$\begin{aligned} v_x &= v \cdot cos\Theta \cdot cos\Psi \\ v_y &= v \cdot cos\Theta \cdot sin\Psi \\ v_z &= v \cdot sin\Theta \end{aligned} \tag{2}$$

$$\frac{dv_x}{d_t} = \frac{dv}{dt} \cdot cos\Theta \cdot cos\Psi - v \cdot sin\Theta \cdot cos\Psi \cdot \frac{d\Theta}{dt} - v \cdot cos\Theta \cdot sin\Psi \cdot \frac{d\Psi}{dt} \tag{3}$$

$$\frac{dv_y}{d_t} = \frac{dv}{dt} \cdot cos\Theta \cdot sin\Psi - v \cdot sin\Theta \cdot sin\Psi \cdot \frac{d\Theta}{dt} + v \cdot cos\Theta \cdot cos\Psi \cdot \frac{d\Psi}{dt} \tag{4}$$

$$\frac{dv_z}{d_t} = \frac{dv}{dt} \cdot sin\Theta \cdot + v \cdot cos\Theta + \frac{d\Theta}{dt} \tag{5}$$

2.4 Path controller, destination controller and solver

The path controller evaluates the position (current state) of the plane and pass the coordinates of the next WP to destination controller. A list of WP (generated path) is obtained from the route generator, see Fig. 4.

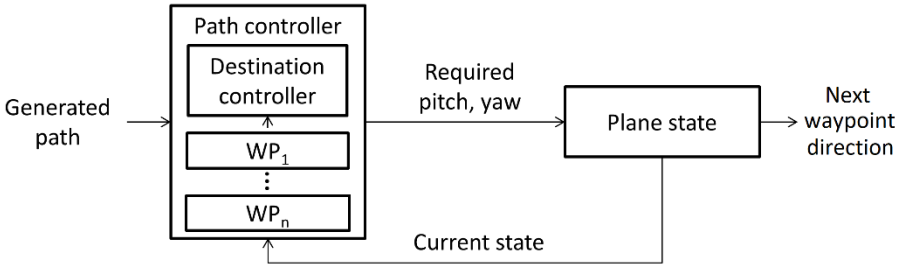

Figure 4: Path controller feedback.

The required state parameters for the plane model (Yaw, $Pitch$ and $\vec{v}$) are computed by the destination controller to get the plane from the current position to the next WP (destination position). The inputs for the controller are the current position of the plane, its current yaw and pitch, and the coordinates of the next WP. Based on this input, the radius vector and required yaw and pitch are computed. The plane is controlled with full manoeuvre (maximum values of yaw and pitch acceleration) and the value of velocity is constant. To avoid control oscillation, the dead zones of yaw and pitch angle tolerance are set.

The ODE model solver is realized by the Runge-Kutta method of order 4(5), for infinite time line generation, from SciPy Python module. The compute function encapsulates the solver for ordering the outputs in Event Calendar part, see code in library "simodes" [19].

2.5 Fire unit and missile model

The generic model of a FU is represented by the probability distribution of time characteristics of fire readiness procedures, by its position, by missile ODE models and a missile quantity. For the purpose of the article the generic model of FU is inspired by the one channel SBAD system SA-6 Gainful [21]. The number of channels stands for the number of simultaneously engageable targets by one FU. The missile ODE model is particularly based on the plane ODE model and destination controller, where the certain destination is the current target position. To simulate a real missile behaviour, the proportional guidance method is implemented accordingly. The

guidance method is defined by the kinematic equations [22], which are implemented into the Python function, see Fig. 5, where is the example of a missile and target trajectory using proportional guidance.

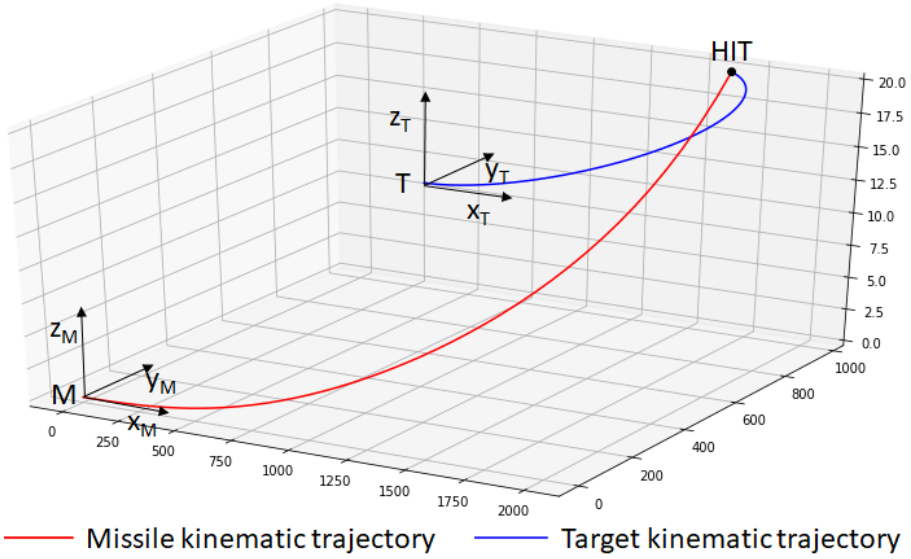

Figure 5: Missile and target kinematic trajectory.

2.6 Radar model

The ODE model of the radar provides essential functionality of 2D radar entity. It is represented by following functions:

- RadarModel – includes the state equation of a model with parameters of acceleration and velocity and defines the angular velocity of rotation in azimuth ω_{az}.
- RadarSeeTarget – calculates radius vector (r) between the radar position and the target, the distance between them and the current radar bearing angle (φ), according to Eq. (6). This function also defines the radar limits (minimum and maximum range) and the angle of view. If all the conditions (target is in the range and in the right direction) are met, the function RadarSeeTarget returns target distance and radar orientation,
- PlaceRadar – a function for radar model positioning, connection to ODE solver and model identification for the Event Calendar. It allows to create multiple different radar models.

$$r = \sqrt{x^2 + y^2}$$
$$\varphi = atan2(y, x) \tag{6}$$

Currently, the radar model do not include the distribution probability of detection and the target Radar Cross Section (RCS). This part is going to be implemented subsequently according to the [23].

2.7 Command post, fire distribution centre model

The CP/FDC model is designed as a centralised type of C2 system with decentralised execution and elemental structure and functionalities. The CP/FDC model ensures three main tasks: AO identification, Target assignment to FU, and Target engagement order issue. The following list includes all the functions in the C2 process (with the entity providing a function in parentheses).

- Air surveillance (RADAR).
- Aerial object detection (RADAR).
- Aerial object ID Assignment (RADAR).
- Aerial object identification (CP/FDC).

- Target assignment to FU (CP/FDC).
- Target engagement order issue (CP/FDC).
- Target engagement execution (FU).
- Target elimination confirmation (FU).

The functions of RADAR and FU can be represented by the results of its ODE models solution or by time (the time delay with uniform or normal distribution). The CP/FDC function is expressed by its time delay.

Each function has a unique name and following arguments:

- addEvent – add simulation state to the event list,
- time – assign simulation time to the event,
- sim – call simulation solver function,
- targetId – assign unique identification number of aerial object, and conduct following actions:
  - sim.Addlog – add the function data to the simulation log,
  - sim.AddEvent – perform/add functionality time delay according to the chosen probability distribution with defined limit values (MIN and MAX).

2.8 Target assignment and queuing theory

In a case of multiple targets or fire units, the Queuing Theory (QT) was applied [13]. According to the Kendall's notation, the system is type *M/M/n/r*. The first *M* stands for the Poisson distribution of incoming targets (λ), second *M* represents the Poisson distribution of service (μ), *n* is a number of FU (parallel channels) and *r* defines the limits of a queue (number of targets or waiting time). The value p_k is a probability of a successful engagement.

$$p_{k+1} = \frac{\lambda}{\mu}\frac{p_k}{n} \Longrightarrow p_0 \left(\frac{\lambda}{\mu}\right)^k \frac{1}{n!\, n^{k-n}}; k = n+1, \dots, n+r \tag{7}$$

Presented solution is type *D/G/n/*∞ type of QT system was used. The incoming targets are deterministically defined (*D*), the C2 system has the normal distribution (*G*), the number of FU is *n* and the queue is infinite (∞). In the case of no vacant FU, the target is waiting in a queue and will be engaged immediately when some FU gets vacant.

2.9 Command and control procedures

The C2 procedures are presented in the form of a state model, which consists of three main entities (blocks): the radar, the CP, and the FU. The State model illustrates the functionalities of entities and their connection, see Fig. 7. The state model of the FU shows the two-channel channels system, which is solved by the QT according to the Subchapter 2.8.

The presented state model is inspired by [5] and can be interpreted in the following way. When simulation begins, the ODE model of a radar starts to survey. When the AO meets the radar parameters (range, azimuth), it is detected, assigned a unique ID number, and the information is forwarded to CP/FDC. CP/FDC ensures the AO IFF and, in the case of a hostile target, it is assigned with a COVER and subsequently an ENGAGEMENT order to the FU 1 or FU 2 according to the QT (if the FU is vacant). Assigned FU conducts the fire readiness procedure. It follows the target and if the target position meets the FU limits (range, visibility, number of missiles), the missile ODE model is created and a missile is launched towards the target. The HIT/MISS evaluation is subsequently provided. If the missile did not hit the target and it still meets the FU limits, the engagement process can be repeated. If AO parameters are not met with the FU limits or the FU is occupied, the FU sends a NO JOY feedback report, which stands for no or impossible activity, and the process is repeated.

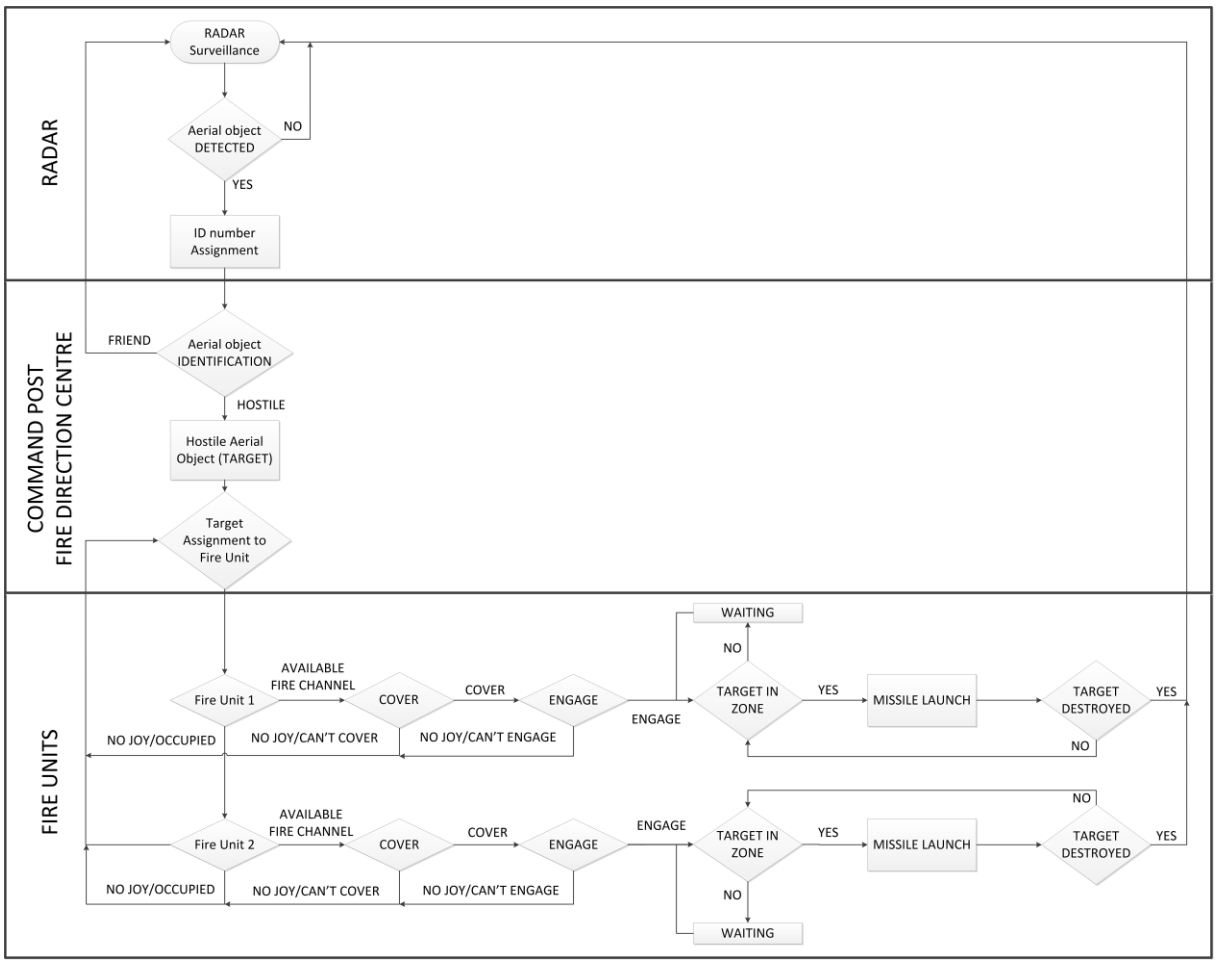

Figure 6: State model diagram of C2 procedures.

3. SCENARIO DESIGN AND EXPERIMENTS RESULT

The presented solution was verified in the following scenarios.

3.1 Scenario one – C2 process evaluation with time models

The first scenario consists of three hostile AOs (targets) and two FU, where the initial time of target detection is randomly generated using Python random library. The target in this scenario is stationary and the scenario serves for the functionality verification and evaluation of the C2 process and QT algorithms.

Table I: Procedures time settings.

C2 procedure name	**Min. time** [s]	**Max. time** [s]
1. AO detection	3	5
2. AO inf. transfer to C2	5	10
3. AO identification	3	5
4. Target assignment to FU	7	15
5. Target is covered by FU	3.5	5
6. Target is irradiated by FU	5	7
7. Target is aimed by FU	1	4
8. Missile is launched	10	30
9. Target is destroyed	3	3

Each C2 procedure step uses the normal probability distribution of time settings according to the Table I. The overall probability distribution of successful target hit for 1000 repetitions is in Fig. 7.

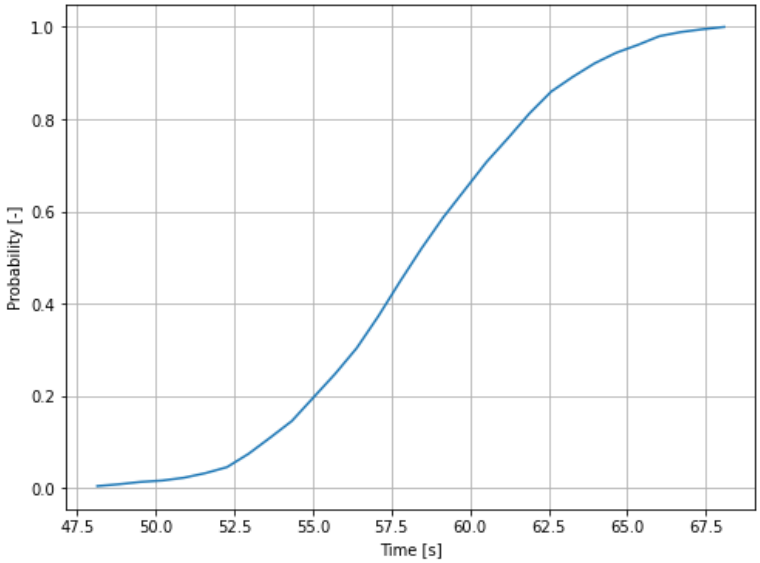

Figure 7: Probability distribution of successful target hit.

3.2 Scenario two – C2 process evaluation with ODE models

This scenario implements RADAR, two FU and three TARGET ODE models with following generic parameters. The generic parameters are for the models and simulation function verification and can be modified based on real entity parameters.

- RADAR – range from 10 to 1500 m, rotation speed 0.2 rad/s.
- FU 1 and FU 2 – position [0 0 0] m, missile with proportional guidance method according to Subchapter 2.3, initial missile velocity 150 m/s.
- TARGET 1 – initial position [1500 0 20] m, initial velocity [10 0 0] m/s, start time t = 0 s.
- TARGET 2 – initial position [1500 0 20] m, initial velocity [10 0 0] m/s, start time t = 10 s.
- TARGET 3 – initial position [1500 0 20] m, initial velocity [10 0 0] m/s, start time t = 20 s.

Targets are time shifted by 10 s and follows the same path: initial position, WP 1: [1000 2 20], WP 2: [500 0 10], final point: [0 0 5]. This scenario represents the situation where flight of three planes follows the same path, where the first plane is the leader and the rest follows him with the time delay. For SBAD units this scenario verifies the target assignment, and possibility of multiple engagement, if there is enough time to engage all the targets. The C2 structure and time setting is identical with scenario one.

3.3 Results of scenario one

Generally, the results fulfil the expectations. ODE models, event calendar and C2 algorithm works properly according to their settings.

The time results of the one iteration of scenario one simulation are in Table II, where all the targets were successfully detected, processed to CP/FDC, identified as hostile and engaged by FU 1 and FU 2. It should be noticed, that the C2 and QT algorithm assign the Target 1 to FU 1 and Target 2 and 3 to FU 2, where the Target 3 is engaged after the Target 2.

Table II: Evaluation of scenario one.

C2 procedure name	**Time AO 1** [s]	**Time AO 2** [s]	**Time AO 3** [s]
1. AO detection	0	0	0
2. AO inf. transfer to C2	3.49	3.64	4.65
3. AO identification	10.78	12.55	13.46
4. Target assignment to FU	15.76 (FU 1)	16.06 (FU 2)	58.19 (FU 2)
5. Target is covered by FU	25.57	25.51	67.99
6. Target is irradiated by FU	30.01	29.61	72.09
7. Target is aimed by FU	36.42	36.11	77.23
8. Missile is launched	38.98	38.17	79.36
9. Target is destroyed	58.84	54.56	96.27

Entire simulation ended in 96.27 s, when the last Target was successfully destroyed.

3.4 Results of scenario two

According to the scenario two settings, see Subchapter 3.2, the ODE models and C2 steps work properly. The time results of each step are in Table III.

Table III: Evaluation of scenario two.

C2 procedure name	**Time AO 1** [s]	**Time AO 2** [s]	**Time AO 3** [s]
1. AO detection	6.14	12.39	29.94
2. AO inf. transfer to C2	9.66	16.45	28.89
3. AO identification	17.97	21.94	36.33
4. Target assignment to FU	21.77 (FU 1)	26.35 (FU 2)	56.58 (FU 1)
5. Target is covered by FU	32.37	37.46	69.26
6. Target is irradiated by FU	36.82	41.64	73.52
7. Target is aimed by FU	42.84	47.74	80.27
8. Missile is launched	46.16	49.09	83.71
9. Target is destroyed	52.74	56.11	89.19

The entire simulation ended in 89.19 s, when the last Target was successfully destroyed. Target 3 was assigned in time 56.58 s to FU 1 according to QT after the destruction of Target 1 in time 52.74 s. It could be also assigned to FU 2, because it was available in time 56.11 s.

All of the output data are logged in the text form with defined structure, where each line describes the time of significant actions for each entity with unique ID during simulation. For example, the time 56.11 s when Target 2 (ID: vcplarmtyh) was successfully destroyed by the missile (ID: tebneilnsj), through times 56.58 s, 69.26 s, 73.52 s, 80.27 s when Target 3 (ID: gqmsnvdyhf) was assigned, covered, irradiated and successfully aimed by the FU 1 in time 83.71 s when the FU 1 launched the missile (ID: gqmsnvdyhf) towards the target, and it is destroyed in 89.19 s. The unique ID code of each entity in simulation is automatically generated and represent the real ID assignment procedures.

4. DISCUSSION

Presented article deals with the design and implementation of experimental and educational simulator for SBAD applications called EXEDAD SIM. The simulation is based on a hybrid simulation and its philosophy reflects several initial requirements.

- First, the design of a missile and target model with kinematic trajectory (flight).
- Second, the C2 procedures, structure and its effective usage for various SBAD task.
- Third, structured, open and easily understandable platform, which can be effectively used for experimental and educational purposes.

The first requirement is realized by Path and Destination Controllers, see Chapter 1 and Subchapter 2.4, which enables the possibility to generate the high number of different tracks, which can be statistically processed in accordance with a simulation given requests.

The second requirement is secured by the variable structure of C2 system where each step/sequence is designed as a separate function, with changeable parameters (time values and probability distribution), see Chapter 1 and Subchapter 2.7. Moreover, the QT procedure for multiple objects was implemented.

The third requirement is fulfilled by the modern Jupyter platform, which combines text descriptors with the Python code and can be executed step by step for easier understanding, see Subchapter 1.3. The given solution, structure and particular parts are described in detail in mainly in Chapter 2. The solution was also verified, tested and evaluated in selected scenarios, see Chapter 3.

Despite the fact that the presented scenarios include the generic values of models, the simulation prove to be functional, valid and the models can be easily replaced and modified according to the user requirements.

5. CONCLUSION AND WAY AHEAD

The main contribution of the article is the SBAD experimental simulator design, we called EXEDAD SIM, where each of the three main pillars (sensors, effectors and C2 system) is modelled in different way and implemented into the overall hybrid simulation. The presented design is unique mainly due to its successful integration of several well-known simulation and modelling techniques (ODE models, DES and QT) with intention to use it in specific military (SBAD) tasks.

The output of the simulator will be used during the mission planning to create various variants of COA for decision-making support. The additional benefit is also its open platform, where all the models can be easily modified and the whole concept is also effectively used in SBAD cadets education and training.

The future work will be focused on the design of more complex scenarios and output data customisation to support machine learning and AI application. The final solution should provide more effective SBAD tactics evaluation and planning and with improved ergonomic user interface it will be also applicable for FDC operators training. For more effective usage we would like to implement optimisation algorithms for sensor and effector networks [24].

ACKNOWLEDGEMENTS

The Czech Republic Ministry of Defence – University of Defence in Brno development program DZRO FVT AIROPS, has supported the work presented in this article. All time values, command and control system structures, tactical and technical data of military systems are generic just for the purpose of the article and they are not classified.

REFERENCES

[1] Pandey, H.; Shukla, V.; Gupta, C. (2020). Aerial war game simulation, *8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)*, 324-331, doi:10.1109/ICRITO48877.2020.9197912

[2] Lopes, H. S.; Lima, R. S.; Leal, F. (2020). Simulation project for logistics of Brazilian soybean exportation, *International Journal of Simulation Modelling*, Vol. 19, No. 4, 571-582, doi:10.2507/IJSIMM19-4-529

[3] Wang, X. (2021). Game-based hybrid particle swarm optimization of job-shop production control, *International Journal of Simulation Modelling*, Vol. 20, No. 2, 398-409, doi:10.2507/IJSIMM20-2-CO9

[4] Gocken, T.; Dosdogru, A. T.; Boru, A.; Gocken, M. (2019). Integrating process plan and part routing using optimization via simulation approach, *International Journal of Simulation Modelling*, Vol. 18, No. 2, 254-266, doi:10.2507/IJSIMM18(2)470
[5] Stary, V.; Farlik, J.; (2020). Aspects of air defence units C2 system modelling and simulation, Mazal, J.; Fagiolini, A.; Vasik, P. (Eds.), *Modelling and Simulation for Autonomous Systems*, Springer, Cham, doi:10.1007/978-3-030-43890-6_28
[6] Stefek, A.; Casar, J.; Stary, V.; Gacho, L. (2021). Air defence command and control system modelling and simulation for war-gaming, *2021 International Conference on Military Technologies (ICMT)*, 4 pages, doi:10.1109/ICMT52455.2021.9502799
[7] Stefek, A.; Casar, J.; Stary, V. (2020). Flight route generator for simulation-supported wargaming, *19th International Conference on Mechatronics – Mechatronika*, 5 pages, doi:10.1109/ME49197.2020.9286646
[8] Hodicky, J.; Prochazka, D. (2017). Challenges in the implementation of autonomous systems into the battlefield, *2017 International Conference on Military Technologies (ICMT)*, 743-747, doi:10.1109/MILTECHS.2017.7988855
[9] Casar, J.; Farlik, J. (2020). The possibilities and usage of missile path mathematical modelling for the utilization in future autonomous air defense systems simulators, Mazal, J.; Fagiolini, A.; Vasik, P. (Eds.), *Modelling and Simulation for Autonomous Systems*, Springer, Cham, doi:10.1007/978-3-030-43890-6_20
[10] Hamtil, I.; Sebela, M.; Stefek, A. (2013). Radar information creation with use of a simulation environment, *IET Radar, Sonar and Navigation*, Vol. 7, No. 4, 333-341, doi:10.1049/iet-rsn.2012.0257
[11] Mažeikis, E. (2017). *NATO Standard APP-6: NATO Joint Military Symbology*, Edition D Version 1, NATO Standardization Office, Brussels
[12] Pietkiewicz, T.; Kawalec, A. (2016). A method of determining the basic belief assignment for combined primary and secondary surveillance radars based on Dezert-Smarandache theory, *17th International Radar Symposium (IRS)*, 6 pages, doi:10.1109/IRS.2016.7497297
[13] Sundarapandian, V. (2009). *Probability, Statistics and Queuing Theory*, 1st edition, PHI Learning, New Delhi
[14] JupyterLab. JupyterLab Computational Environment, from *https://github.com/jupyterlab/jupyterlab*, accessed on 01-11-2021
[15] Bisong, E. (2019). Google Colaboratory, Bisong, E. (Ed.), *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*, Apress, Berkeley, 59-64, doi:10.1007/978-1-4842-4470-8_7
[16] Staubitz, T.; Klement, H.; Teusner, R.; Renz, J.; Meinel, C. (2016). CodeOcean – a versatile platform for practical programming exercises in online environments, *2016 IEEE Global Engineering Education Conference*, 314-323, doi:10.1109/EDUCON.2016.7474573
[17] Lasser, J. (2020). Creating an executable paper is a journey through Open Science, *Communications Physics*, Vol. 3, Paper 143, 5 pages, doi:10.1038/s42005-020-00403-4
[18] Perkel, J. M. (2018). By Jupyter, it all make sense (Online: Why Jupyter is data scientists' computational notebook of choice), *Nature*, Vol. 563, No. 7729, 145-146, doi:10.1038/d41586-018-07196-1
[19] Simodes 0.7. Python Library, from *https://pypi.org/project/simodes/*, accessed on 01-11-2021
[20] Naushad, F.; Srikanth, K. S. (2017). In-flight route replanning and survivability analysis for a fighter aircraft, *2017 International Conference on Recent Advances in Electronics and Communication Technology*, 11-16, doi:10.1109/ICRAECT.2017.27
[21] Macfadzean, R. H. M. (2000). *Surface-Based Air Defense System Analysis*, illustrated edition, Artech Print on Demand, London
[22] Siouris, G. M. (2004). *Missile Guidance and Control Systems*, Springer, New York
[23] Knott, E. F.; Schaeffer, J. F.; Tulley, M. T. (2004). *Radar Cross Section*, 2nd edition, SciTech Publishing, New York
[24] Fu, X.; Yao, H.; Yang, Y. (2021). Modeling and optimizing the cascading robustness of multisink wireless sensor networks, *IEEE Transactions on Reliability*, Vol. 70, No. 1, 121-133, doi:10.1109/TR.2020.3024797