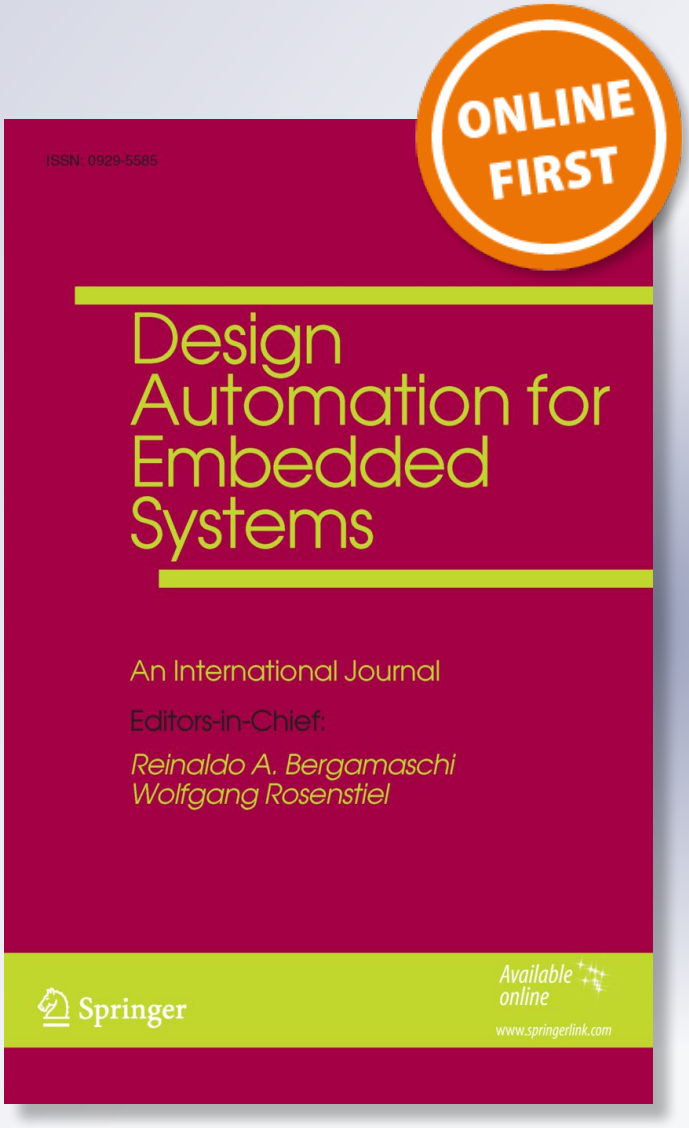
Analysis of design strategies for unmanned aerial vehicles using co-simulation

José de Sousa Barros, Thyago Oliveira Freitas, Vivek Nigam & Alisson V. Brito

Design Automation for Embedded Systems
An International Journal

ISSN 0929-5585

Des Autom Embed Syst
DOI 10.1007/s10617-017-9190-z



Springer

Des Autom Embed Syst
DOI 10.1007/s10617-017-9190-z

Analysis of design strategies for unmanned aerial vehicles using co-simulation

José de Sousa Barros[1] · **Thyago Oliveira Freitas**[1] · **Vivek Nigam**[1,2] · **Alisson V. Brito**[1]

Received: 16 March 2017 / Accepted: 13 September 2017

Abstract Designing critical embedded systems, like UAVs is not a trivial task because it brings the challenge of dealing with the uncertainty that is inherent to this type of systems, e.g., winds, GPS uncertainty, etc. Simulation and verification tools that provide a level of confidence can help design such systems and increase the safety of specified cyber-physical systems before deployment. This paper presents a framework for evaluating flight strategies of UAVs. Our framework is constructed by integrating, using high-level architecture, Ptolemy, a high level specification tool, and SITL/Ardupilot, a domain specific UAV simulator. It allows to evaluate flight strategies under the presence of uncertainty, such as winds, with a level of confidence by constructing a sufficiently large number of simulations. Its effectiveness is demonstrated by testing two different flight strategies in two scenarios under different wind intensities. We measure the flight quality providing quantitative information about the quality of the tested flight strategy, such as distance traveled, with a confidence of 95% and error of 8%.

Keywords Co-simulation · UAV · Testing

✉ Alisson V. Brito
alisson@ci.ufpb.br

José de Sousa Barros
caianajsb@gmail.com

Thyago Oliveira Freitas
thyago.oliveira@eng.ci.ufpb.br

Vivek Nigam
vivek@ci.ufpb.br

1 Federal University of Paraiba (UFPB), João Pessoa, Brazil

2 Fortiss, Munich, Germany

Published online: 21 September 2017

1 Introduction

Unmanned Aerial Vehicles (UAVs) are aircrafts capable of performing flight missions without the presence of an on board crew controlling the vehicle. Most of these vehicles are radio-controlled by a pilot on the ground. In the current context of aviation technologies, the use of this type of vehicles is in ascendancy, arousing interest from companies, institutions and individuals for many different applications [1–3].

Designing critical embedded systems, like UAVs is not a trivial task because it brings the challenge of dealing with the uncertainty that is inherent to this type of systems, e.g., winds, GPS uncertainty, etc. Complex embedded systems are generally heterogeneous. It means that the same model may be composed of separate modules in relation to programming languages, abstraction levels and combination of software and hardware. Therefore, the design of this kind of system requires a number of specific tools for modeling and simulation. The Ptolemy project proposes a tool that allows design and simulation of heterogeneous embedded systems in a single environment [4].

Specifically about the UAVs, there is a lack of tools which allows the designer not only plan the flight, but integrate it with the specific mission and evaluate the adopted strategy in hazardous environments. Even though, there are many consolidated tools specialized in simulating and evaluating specific features (see Related Work section for more details.) Thus, a solution might be the integration of different simulators in an unique framework.

This work is based on the hypothesis that there is a combination of tools that are effective in solving part of the problem, which can be integrated to result in a system capable of generating information that is even more relevant to the designers. Existing tools are often lacking in evaluating different flight strategies. For example, for a plant monitoring application, where each point must be monitored with a certain frequency, what strategy might be adopted for the UAV always keep the various points under surveillance and at the same time maintain a safe battery level, even in difficult conditions such as wind or loss of contact with the base? When we talk about strategy here, we mean planning what the UAV should do in each situation. What should it do if the battery reaches a certain critical limit? What to do if the wind takes the UAV to an undesirable point?

We presented in our previous work [5] a proposal for evaluating UAV flight strategies. However, in the experiments carried out in the proposal, only one flight was observed for each modeled strategy. This paper considerably extends our previous work by developing the machinery by allowing to evaluate a flight strategy by automatically generating multiple executions increasing thus the confidence of our experimental results. This paper adds more than 190 h of simulations distributed over more than 150 experiments.

As presented in previous work, we use the High Level Architecture [6] to modularly integrate the components of our framework. HLA specifies an infrastructure for time management in the various simulators, called Runtime Infrastructure, or RTI. It enables the integration of heterogeneous systems in a synchronous and transparent manner, taking advantage of all pre-existing systems. In particular, we used HLA to integrate the modeling tool Ptolemy and the domain specific machinery SITL/ArduPilot [7] for UAVs. The use of SITL/ArduPilot implements a realistic controller used in many UAVs, called Ardupilot and thus their physical behavior, but lacks a high-level specification language. By integrating SITL/ArduPilot with Ptolemy, we are able to specify flight strategies in a high-level language and still use the physical model of UAVs provided by SITL/ArduPilot.

Our efforts are towards tackling the complexity of designing CPS by providing a reliable infrastructure to simulate and evaluate flight scenarios. Thus, it is necessary cooperation

between different tools in a simple and synchronized manner. As the main contribution and scientific relevance of this work, we may highlight the development and evaluation of a simulator of UAV applications which analyzes strategies (using Ptolemy) and physical aspects of UAV (using SITL/ArduPilot), integrated by HLA.

The paper is organized as follows. Section 2 presents some related works. In Section 3 the basic fundamentals of HLA are shortly described. The proposed framework is described in Section 4, followed by the experiments and results presented, in Sections 5 and 6, respectively. Some final considerations are described in Section 7.

2 Related work

There are several tools able to assist the design and validation of embedded systems [4,8,9] and, therefore the use of these tools should not be overlooked. Nevertheless, simulation using only one tool cannot always meet all the characteristics of embedded systems because these systems are often complex and heterogeneous. This issue is gaining attention from researchers.

Other work [10] proposes a new approach to co-simulate hardware and software with the concept of a bridge component between two simulators. In that approach the Giano and ModelSim simulators are specifically integrated. Unlike our work that proposes the integration between any simulators via a consolidated standard. The authors use the concept of a bridge component to interface hardware with software parts, all in a solution specific to a scenario, while our approach is based on HLA [6], a consolidated standard, which enables the integration of any other HLA-compliant component.

It was also proposed a method based on Matlab/Simulink, which consists of modeling, simulation, verification and code generation [11]. The software codes and embedded systems prototyping can be checked step-by-step using co-simulation between Matlab and Simulink. The tool proposed in this work is based on only open-source tools and standards.

Co-simulation is also used in [12], which presents a software platform that can be used to design embedded system composed by multiple processors. That solution is based on the interaction of a software running in a processor model and the hardware device simulated by SystemC. This platform can perform virtual prototyping of new hardware devices, unlike our proposal, where different simulators are integrated to allow the simulation of UAV flights strategies.

In [13], it is presented a collaborative approach that allows engineers from different areas the construction of individual models in the most appropriate ways. It also allows the co-simulation of these models in a common platform. The approach was performed using Crescendo[1] technology, which allows the definition and simulation of composite Discrete Event models expressed in VDM notation (Vienna Development Method) and Continuous Time models expressed using the 20-sim Framework.[2] Crescendo allows models running in different simulators, transferring data and managing simulation time. Differently, our approach relies on HLA to transfer data and manage synchronization among all simulators.

A distributed simulation platform using HLA for embedded systems projects is proposed by [14], which presents the experimental results of five different scenarios, which integrate five different simulation tools: Ptolemy II, SystemC, Omnet++, Veins, Stage and physical robots. The experiments were successfully carried out in the application of wireless sensor

[1] http://www.crescendotool.org.

[2] http://www.20sim.com.

networks, energy estimation in circuit design, robotic simulation and co-simulation of real robots. That work also use Ptolemy and HLA, but in this presented work we also integrate with SITL/ArduPilot to simulate the behavior of UAVs.

A framework for distributed simulation of cyber-physical systems (CPS) is presented in [15], which uses Ptolemy and HLA. Ptolemy extensions are presented for interaction with HLA and the approach is demonstrated in a simulation of a flight control system. It differs from our proposal because here the integration with the SITL/ArduPilot is performed to represent the behavior of UAVs, whereas they use Ptolemy with no details of the physical environment.

A modeling platform for the design of cyber-physical systems is presented in [16]. A case study with Unmanned Aerial Vehicles is modeled and simulated using Ptolemy. The authors afirm that adding more detail to the physical processes would bring credibility to the project. Our approach fills this gap by integrating Ptolemy with the flight plan simulator SITL/Ardupilot which adds the details of the the physical environment.

The design of systems based on UAVs relies on tools that are capable of delivering details of the vehicle itself and also from its interaction with the environment. In addition, there are uncertainties in the environment where the system will act to be taken into account during the design, because they somehow influence the system operation.

In the previous works [14,17] is demonstrated the integration of Ptolemy with HLA to allow the simulation of heterogeneous systems in a distributed way. In addition, some simulators are integrated with Ptolemy through the HLA, but no integration is performed with any UAV simulator. This is supplied by work [5], which presents the integration of Ptolemy with SITL/Ardupilot. In the current work, uncertainties related to the occurrence of wind are added to the flight environment. Also, several simulations are carried out (more than 190 h of simulation). In addition, statistical results are presented for each simulated flight configuration. Here it is demonstrated that one strategy, developed by the designer, can be replaced by another through the use of visual components. This characteristic is not found in other works. In addition, all the tools used might be easily replaced or new ones added through the HLA.

In this scenario, the goal of this work is to build a simulation environment where it is possible to analyze UAV flights strategies using co-simulation. The idea is to take advantage of each tool and use it in order to analyze strategies before the flight itself. For this, we integrate Ptolemy [4] with SITL/ArduPilot [7] using HLA [6] as middleware.

3 High level architecture

The HLA is a standard of the Institute of Electrical and Electronic Engineers (IEEE), developed by Simulation Interoperability Standards Organization (SISO). While initially it was not an open standard, it was later recognized and adopted by the Object Management Group (OMG) and IEEE.

There are several standards based on distributed computing, such as SIMNET, Distributed Interactive Simulation (DIS), Service Oriented Architecture (SOA), Data Distribution Service (DDS), HLA, among others. HLA was chosen as middleware to integrate distributed heterogeneous devices because it manages both, data and synchronization, and allows the interoperability and composition of the widest possible range of platforms.

HLA is defined by three documents: the first deals with the general framework and main rules [6], the second concerns the specification of the interface between the simulator and

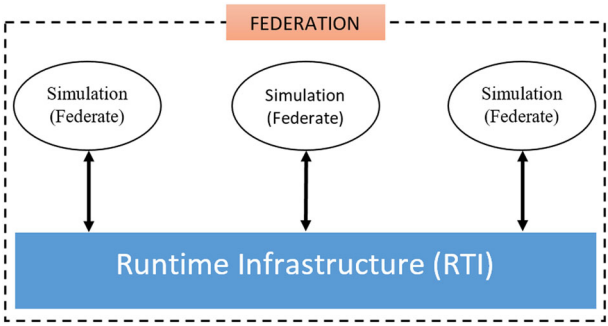


Fig. 1 Architecture of a federation

the HLA [18] and the third is the model for data specification (OMT) transferred between the simulators [19].

The main HLA characteristics are defined under the leadership of Defence Modelling and Simulation Office (DMSO) to support reuse and interoperability. Interoperability is a term that covers more than just send and receive data, it also allows multiple systems to work together. However, the systems must operate in such a way that they can achieve a goal together through collaboration.

In HLA architecture (see Fig. 1), the set of various interoperating systems within a domain is called Federation. Each member of a federation is called Federate [6] . The Federates are registered and managed through a Runtime Infrastructure (RTI) , as can be shown in Fig. 1.

Each Federate is locally associated with a RTI Ambassador (RTIA) process via TCP socket. Messages between RTIA and RTI Gateway (RTIG) are exchanged through a TCP/IP network protocol in order to perform the RTI services in a distributed manner. The RTIG is the central point in the architecture. It manages the data exchanging and synchronization between all Federates in a Federation.

By making use of the HLA, the proposed tool allows integration with other simulation tools or even with a physical UAV if necessary. This is because for HLA each component in the simulation, physical or virtual, must implement the services of the HLA Interface and then it is considered as a Federate. So it is not necessary for a tool to know the details of the others. In addition, HLA is responsible to centralize and manage the time advance of each federate in the simulation. This allows the simulation tools to be responsible only for representing the details in their context, not concerning about time synchronization algorithms to interact with the other tools. The interface among each tool and HLA should be implemented only once. In our case, in used the Ptolemy Federate implemented in our lab [17] and implemented for this project a Federate to connect SITL to the environment.

4 A framework for analysis of strategies for UAVs

The proposed simulation framework consists of two parts (as presented in Fig. 2), one part is responsible for representing the flight environment (using SITL/Ardupilot), and the other part is responsible for the definition of the flight strategy that will be executed by the UAV (using Ptolemy). Synchronization and communication between the parts is made using High-Level Architecture - HLA.

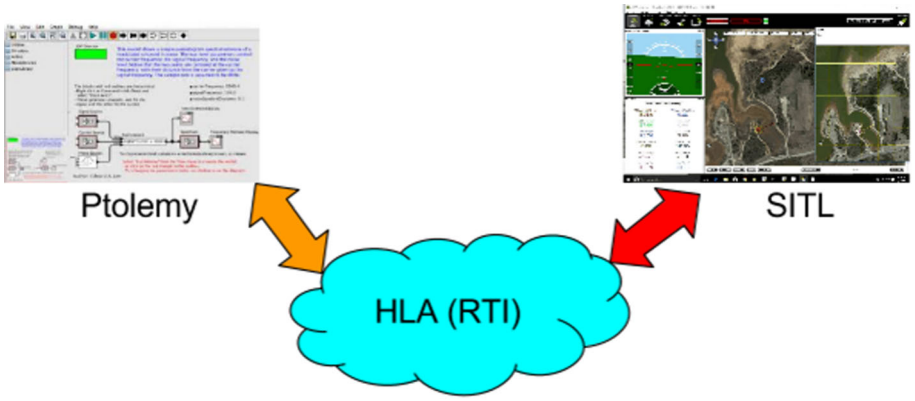


Fig. 2 General architecture of the framework

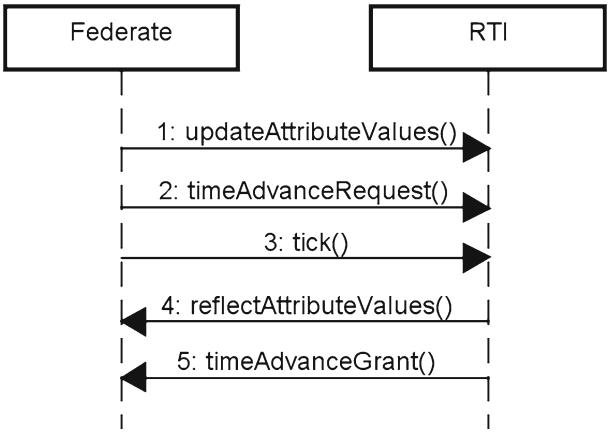


Fig. 3 Interaction of Federate and RTI

The integration between the simulators follows the idea that each simulator must be a Federate in an HLA Federation. This can be seen in Fig. 1, where the architecture of a federation is presented. Thus, as a Federate is responsible for sending and receiving data according to the HLA standard. It uses JCerti (http://savannah.nongnu.org/projects/certi) as HLA implementation, and the implementation to enable the SITL/Ardupilot to be a Federate was developed in this project and uses PyHLA (www.nongnu.org/certi/PyHLA).

In a simulation, the exchange of data between the simulators happens through the invocation of the services specified by the HLA, implemented in the Federate and in the RTI. As shown in Fig. 3, when a Federate needs to publish new data to other Federate, it makes a call to the *updateAttributeValues* method in the RTI and then requests the advancement of its local time with the call to the method *timeAdvanceRequest*. Then the Federate announce the RTI, through the call to the *tick* method, that it is waiting for a response. The RTI, in turn, evaluates whether the request can be grant and, if so, updates the Federate with the last receive data by invoking the *reflectAttributeValues* method, and then grants the time advance With the call to the *timeAdvanceGrant* service in the Federate. This approach is also very versatile, due to encapsulation of each simulator details by HLA. The effort to add a new

Fig. 4 Flight environment running on SITL/Ardupilot

simulator would be to implement a new Federate to it, or reuse an existing Federate (eg. Matlab has HLA support), and adapt it to exchange messages in same format as we do in our tool chain.

In the strategy configuration module developed in Ptolemy, communication with the HLA is performed by the actors *HlaManager, HlaSubscriber* and *HlaPublisher*. They encapsulate calls to services specified by the HLA. *HlaManager* is the Ptolemy federate implementation and can be viewed in Fig. 5, where it appears with the HLA acronym and the name *producer*. It is responsible for managing the advancement of time between Ptolemy and HLA. *HlaPublisher* is responsible for publishing to RTI any information that originates in Ptolemy and should be forwarded to other federates.

This actor is also presented in Fig. 5 where it appears with the name *goto*. It is important to note that this name must be the same as the attribute of registered in the configuration file (see Code 2). In this module, the class declared for Ptolemy was named *robot*. It is presented in the Code 2 and one of its attributes is *goto* representing the command to change the position of the UAV.

In turn, *HlaSubscriber* is responsible for receiving data published in the RTI. The data of interest are those related to the battery charge and the location of the UAV, which are published in the RTI by the SITL. Therefore, in this module there are two actors of type *HlaSubscriber*, one named *battery* to receive the battery level, and another one named *gps* to receive the location of the UAV in Global Positioning System (GPS) format.

Figure 4 shows the module responsible for representing the flight environment in SITL/Ardupilot. It is possible to see that it uses satellite image and maps (from Google Maps) and produces accurate values of UAV compared to real flight environments, like speed, distances and power consumption.

A quadcopter was used in the simulations, and for this type of vehicle the SITL does not support to add the occurrence of wind in the flight environment. Therefore, it was necessary to implement a way to emulate the occurrence of wind (see Code 1). This code is executed by the SITL/Ardupilot Federate and it runs once at each simulation cycle. In this way, the uncertainties of the flight environment were simulated. In line 1 of the presented code a uniform lottery is made between 1 and 100, this means that each number of the range considered has the same possibility of being chosen. When the number drawn is less than or

equal to the number set as the chance of wind occurring, it means that a wind must occur in the flight environment and the next step is to know in which direction it should blow.

Since the direction the wind blows is also uncertain, in line 2 a new draw is made between numbers one and four, 1 for north, 2 for south, 3 for east, and 4 for west. In addition, another uncertainty considered is the intensity that the wind must blow, so in line 4, a new draw is carried out and its result is interpreted as follows: 1 means wind with weak intensity, 2 wind with medium intensity and 3 strong intensity.

This random process is performed at each iteration of the simulation. Then the command with simulated wind interference is sent to the SITL flight environment via the function available in the DroneKit API, *send_ned_velocity()*. This function sends a command to the UAV that changes its position, which emulate the effects of wind against it. In this way, a movement is performed that represents the influence of the wind on the UAV. After that, the targeting command that was published in the RTI by the strategy used in Ptolemy is also sent to the UAV by calling the *send_ned_velocity()* function.

Code 1 Implementation of wind occurrence.

```
1  if random.uniform(1,100) <= chance:
2    direction = random.randint(1,4)
3    # 1–weak, 2–medium, 3–strong
4    intensity = random.randint(1,3)
5    if direction == 1:
6      send_ned_velocity(NORTH,0,0,intensity)
7    elif direction == 2:
8      send_ned_velocity(SOUTH,0,0,intensity)
9    elif direction == 3:
10     send_ned_velocity(0,EAST,0,intensity)
11   elif direction == 4:
12     send_ned_velocity(0,WEST,0,intensity)
```

The module responsible for setting a strategy can be seen in Fig. 5. Actors *StrategyA* and *StrategyB* are presented in this figure. The model is configured to use the connected strategy during the flight simulation. In this case, the *StrategyA* is in use but could be easily replaced by *StrategyB*, as presented in Section 5.

Although Fig. 5 presents only two actors strategies, others may be added through the creation of new actors that implement them. The design of a new strategy is done by implementing a new actor in Ptolemy using Java. To create a new actor it a new Java class must be implemented, which inherits a *TypedAtomicActor* (or other existing Ptolemy actor). After that, input and output must be defined. Finally the *fire()* method of the inherited super class *Actor* should be overwritten with the implementation of the new strategy. This method is responsible for reading the information from the input ports, for executing the strategy and for sending the action through output ports. The *fire()* method of each actor is invoked by Director based on relations among actors, resulted from a schedule algorithm (in our case, Discrete Event).

For the experiments, a surveillance scenario was chosen. In this scenario, a list of location points are passed to the UAV, which should visit all of them continuously. The UAV must take a picture of the points every time it flight over it. No point can stay more than a established time without being visited. Also, the UAV must never flight out of battery under risk to fall down. Thus, the strategy must define battery level as high priority requisite.

In *StrategyA* the UAV visits the target points in the order they are registered, regardless of the distance between the current position of the UAV and the target location. This means that even if the first registered location is far from the current UAV position, and there is

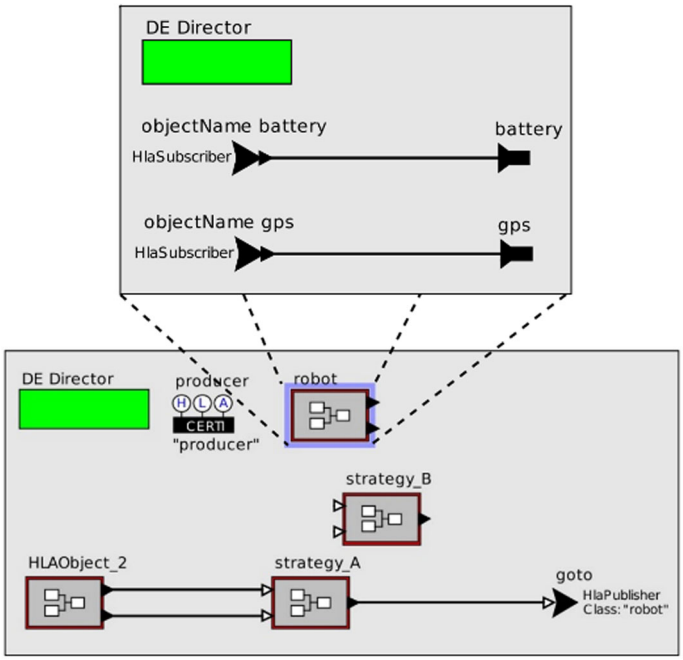


Fig. 5 Ptolemy model configured with StrategyA

another nearest point to be visited, the UAV will not take this under consideration and will visit initial programmed location.

Differently, in *StrategyB* the UAV visits the target points not considering the order they were registered, but the distance between the current position of the UAV and the target point. This means that the closest point to UAV will be visited first, followed by the other points ordered by the distance to UAV at each instant.

4.1 Data modeling

When using HLA, the Federates exchange data in the form of objects defined following the Object Model Template (OMT) from HLA [19], which is specified in an specific file common to all Federation and present at each machine.

As presented in Fig. 5, the actor used was one presented in [20]. The HLA actor has a port for each possible data to be exchanged via HLA. In our approach the ports dedicated to transfer the ID of the UAV (for future usage of multiple UAVs), battery level and position were used for data exchanging, plus one port for sending commands (called "goto"). Through this last port the strategy actor sends to the UAV which movement it should make at each instant. The specification of the data model can be seen in Code 2.

Code 2 Data model used by HLA

```
1  (FED
2    (Federation TestFed)
3    (FEDversion v1.3)
4    (spaces)
5    (objects
```

```
    (class ObjectRoot
      (attribute privilegeToDelete reliable timestamp)
      (class RTIprivate)
      (class robot
       (attribute id reliable timestamp)
       (attribute battery reliable timestamp)
       (attribute position reliable timestamp)
       (attribute goto reliable timestamp)))
  )
)
```

5 Experiments

In our experimentation scenarios, a flight strategy is a set of actions that are be performed by the UAV in order to photograph specific points that need to be visited. One strategy may be to visit and photograph the points in the order in which they were registered, or visit the points based on the time of the last visit and the distance from the UAV to the next target point.

In order to evaluate which strategy is more efficient, the two strategies were executed in two different scenarios that differ only by the size of the area delimited by the points visited by the UAV. The same five points to be visited were set for all scenarios, as well as an initial point for takeoff and landing of UAV.

Considering a better reliability in the obtained results, different probabilities of wind occurrences were simulated for both scenarios: 0, 2.5, 5, 8.3, 12.5 and 25%. For each probability, 153 simulations were performed for strategy A and 153 simulations for strategy B. This means that 918 (153 executions of 6 probabilities) simulations were performed using strategy A and the same using strategy B, totaling 1,836 simulations for Scenario 01. The same experiment was repeated for Scenario 02. The sum of the simulations performed in Scenario 01 and Scenario 02 totaled 3,672 flight simulations. The confidence level calculated for the sample size (153 simulated flights) was 95% and the margin of error was 8%.

For each probability of wind occurrence the computer took about 8 h to execute the 153 simulations. This means that Scenario 01 consumed about 96 h of simulation, 48 h for strategy A and 48 h for strategy B. The same time was required to simulate both strategies in Scenario 02, which means 192 h of Simulation for the whole case study presented in this paper.

For each simulation the UAV started with the battery charge at 100%. It takes off and starts the execution of several iterations until reaching the end of the simulation. The process of completing a simulation is triggered whenever the charge level reaches 20%, at which point the UAV attempts to return to the base station and land. But, if the charge level reaches 5%, it lands immediately regardless its actual localization, also terminating the simulation.

Travel Time is implicitly considered by the battery level. In SITL simulation, the faster the UAV travels, the more power will be consumed. For simplification, all experiments were performed with the UAV travelling with the same speed.

In the Fig. 6 is presented the designated points for the UAV, where the grid informs the latitude and longitude of them. Although the arrangement is similar, the map in Fig. 6b has a larger area, allowing us to analyze its impact on the success of the strategies.

Both scenarios have the same amount of target points and are very similar. The basic difference between them is the size of the monitored area. The path in Scenario 01 has 884.14 meters and a total area of 28,577.12 square meters, and Scenario 02 has a path of 1337,48 meters long and a total area of 62,035.31 square meters. In all the simulations the UAV started the surveillance from the base station, which was established in the same location

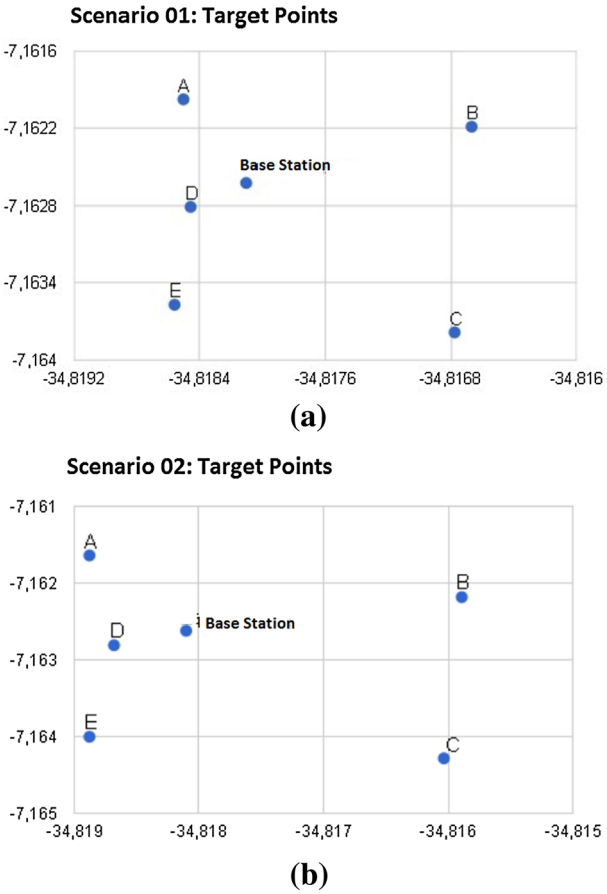


Fig. 6 Target points. **a** Target points in Scenario 01. **b** Target points in Scenario 02

Table 1 Distance in meters between points in Scenario 01

From/to	Base	A	B	C	D	E
Base	0.00	87.70	166.33	199.62	45.06	121.42
A	87.60	0.00	204.57	284.57	97.49	185.94
B	166.33	204.57	0.00	186.45	210.70	263.89
C	199.62	284.74	186.45	0.00	217.98	198.90
D	45.06	97.49	210.70	217.98	0.00	89.20
E	121.42	185.94	263.89	198.90	89.20	0.00

for both scenarios. In Tables 1 and 2 are presented the distance between the target points for Scenario 01 and 02.

6 Results

The main data related to the UAV flights were collected during each simulation in order to compare how the strategies behaved in each situation.

Table 2 Distance in meters between points in Scenario 02

From/to	Base	A	B	C	D	E
Base	0.00	143.09	249.48	299.80	67.64	182.52
A	143.09	0.00	336.01	440.47	138.31	275.72
B	249.48	336.01	0.00	245.64	316.69	392.22
C	299.80	440.47	245.64	0.00	339.30	315.87
D	67.64	138.31	316.69	339.30	0.00	140.82
E	182.52	275.72	392.22	315.87	315.87	0.00

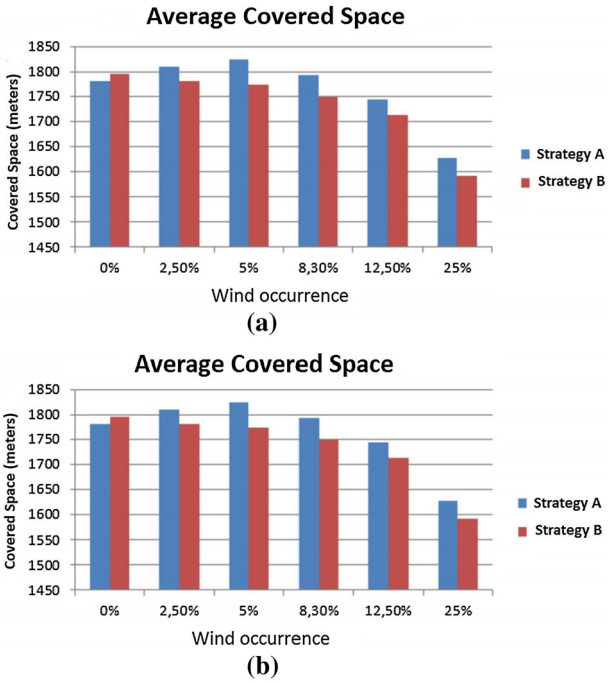


Fig. 7 Average distance in meters traveled by the UAV. **a** Average distance traveled in Scenario 01. **b** Average distance traveled in Scenario 02.

It is also possible to analyze the strategy that allowed a better use of the routes. Figure 7, shows that in the existence of wind, Strategy B achieved better performance, and in the absence, Strategy A was the most satisfactory because it could accomplish the mission travelling shorter distances.

In addition to the distance traveled criteria, it was also analyzed which strategy resulted in a greater amount of obtained pictures. The more captured photos, the better is the coverage of the surveillance system. The data in Fig. 8 show that the average of pictures captured in Strategy B was higher compared to Strategy A, and this occurred even without the occurrence of wind. Despite Strategy A be more efficient with respect to distance traveled, Strategy B was more efficient regarding the amount of captured photos.

However, as important as traversing all points is to analyze how often the UAV returns to the base at the end of the mission. When the battery is at 20% charge, the vehicle will attempt to return to the base station to land, and if it reaches 5% before that happens, it will

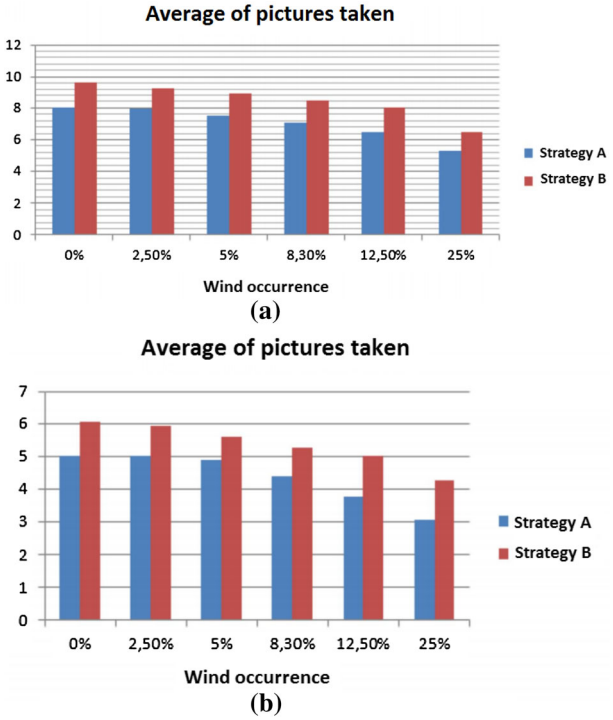


Fig. 8 Average pictures taken by the UAV. **a** Average pictures taken in Scenario 01. **b** Average pictures taken in Scenario 02

land, regardless whether it has reached the target or not. The Fig. 9 shows that in this case, Strategy A achieved better results.

As expected, the presence of wind influenced the number of points visited, and in turn, the number of photos taken from the points of interest. As the performances of the strategies are evaluated, it is simple to analyze which one obtained the better results. In Fig. 8, it is possible to see that even with the higher level of uncertainty through the addition of wind, and larger area covered by the UAV, Strategy B resulted more pictures taken, with about 17.9% more pictures than Strategey A in Scenario 01, and 27.93% more pictures in Scenario 02.

From this result, we observe that in most cases Strategy A achieved a mean distance traveled greater than Strategy B. Except in Scenario 01 with 0% of wind occurrence and in Scenario 02 with 25% wind occurrence, Strategy A presented an average lower than Strategy B. In terms pictures captured, Strategy B was better than the Strategy A in all situations of occurrence of wind in both scenarios, despite the fact that the UAV has traveled less than with Strategy A, which means that Strategy B was more efficient.

In relation to the battery consumption, the remaining battery charge level at the time of landing for Strategy A than Strategy B in cases with 0% and 2.5% possibility of wind in both scenarios. For the other cases, the remaining charge level at the time of landing was the same for both strategies.

For Scenario 01, it is also possible to notice that both Strategy A and Strategy B were able to return to base in most of the cases, with a short reduction to 87.58% of the cases with Strategy B with occurrence of wind of 12.5%. But, when considering the Scenario 02,

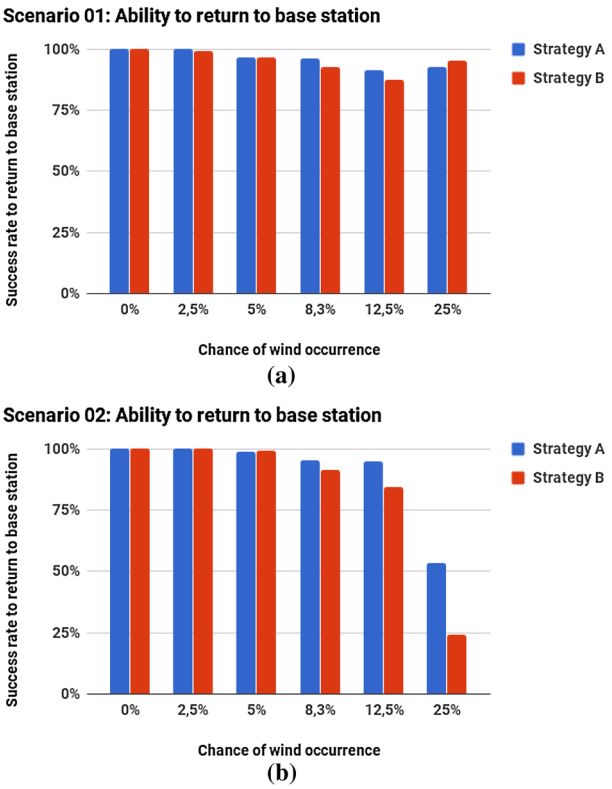


Fig. 9 Average number of times the UAV returned to base. **a** Scenario 01. **b** Scenario 02

which presents a longer path than Scenario 01, Strategy B presented a much slower result than Strategy A, with a return to base rate of 24.18%, compared to 53.59% for Strategy A. In practice, this means that the UAV should be searched often, with risk of be lost, or land over inappropriate places such as power lines, cars or people. This shows the relevance of analysis like the presented in this work, when considering critical systems like UAVs.

7 Final considerations

The proposed environment presented in this work enables simulation of UAVs using different strategies. This allows the designer to analyze various strategies and select the most suitable for each context. This should aid the design of critical cyber-physical systems, specifically to validate flight strategies for UAV systems.

Using co-simulation, it was possible to obtain results closer to reality, thus more efficient and safe strategies can be developed and tested. This approach follows the idea that complex systems can be better modeled and tested when integrating different simulators, joining the best of each one in an unique environment.

The environment is formed by the integration of two simulators. One is responsible for the configuration of the flight strategies (Ptolemy), the other is responsible for representing the

flight environment and the telemetry of the UAV (SITL/ArduPilot). Communication between the simulators was made using High-level Architecture (HLA).

At the end, the most important result for a surveillance UAV is to visit the most number of points (and take pictures) as possible and return safely to base. Thus, with the presented resulted, we demonstrated that our simulation tool was able to evaluate two different strategies showing deeper characterizations. For this experiment, it is possible see that Strategy B could take more pictures and did it travelling shorter distances than Strategy A for both scenarios. But, it was also demonstrated that Strategy B has a lower ability to return to base than Strategy A in some situations from Scenario 01 and 02. This can be a critical limitation. Not return to base might mean landing in an unsafe place, or even worst, hit somebody on the ground and cause serious damages. This risk demands tools not only to simulate dynamics and routes (e.g. SITL) or to simulate algorithms (e.g. Ptolemy), but to simulate both in an integrated way to extract the best of both worlds. That is the main contribution of this work.

As further work, new strategies with other flight algorithms should be implemented to compose a library of strategies ready for use in future simulations. Furthermore, it is expected that some of these new strategies take into account the scenarios where multiple UAVs work together in specific missions. Also, we plan to conduct a greater number of simulations to further increase the level of confidence and reduce the margin of error. Finally, we will make comparisons between the results obtained by simulation with real systems.

Although there are many works focusing on the development of simulators and logic analyzers applied to UAVs, this work stands out by the approach of joining different simulators in a single environment, using tools that have advanced very separately, but when united can achieve even better results. In addition, this work is innovative because it focuses not on the test of algorithms, but on the analysis of the efficiency of the strategy adopted by them to reach their objectives in scenarios as close to reality as possible. A limitation of this work is still the execution time of the simulations, which can be reduced by adding more machines in a distributed simulation, since the HLA supports this. Another limitation is the impossibility of simulating scenarios with multiple UAVs and moving targets. These limitations are being worked out by us and should be presented in future work.

References

1. Luo C, McClean S, Parr G, Teacy L, De Nardi R (2013) UAV position estimation and collision avoidance using the extended Kalman filter. IEEE Trans Veh Technol 62(6):2749–2762
2. Lam T, Boschloo H, Mulder M, van Paassen M (2009) Artificial force field for haptic feedback in UAV teleoperation. IEEE Trans Syst Man Cybern Part A Syst Hum 39(6):1316–1330
3. Teacy W, Nie J, McClean S, Parr G (2010) Maintaining connectivity in UAV swarm sensing. In: IEEE GLOBECOM Workshops (GC Wkshps), December 2010, pp 1771–1776
4. Ptolemaeus C (ed) (2014) System design, modeling, and simulation using Ptolemy II. Ptolemy.org. [Online]. http://ptolemy.org/books/Systems
5. de Sousa Barros J, Oliveira T, Nigam V, Brito AV (2016) A framework for the analysis of UAV strategies using co-simulation. In: VI Brazilian symposium on computing systems engineering (SBESC), November 2016, pp 9–15
6. IEEE standard for modeling and simulation (M&S) high level architecture (HLA)—framework and rules. In: IEEE Std 1516–2010 (Revision of IEEE Std 1516–2000)
7. ArduPilot Dev Team (2016) SITL/Ardupilot Simulator (Software in the Loop). http://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html
8. Forin A, Neekzad B, Lynch NL (2006) Giano: the two-headed system simulator. Microsoft Research, Tech. Rep. MSR-TR-2006-130, September 2006. http://research.microsoft.com/apps/pubs/default.aspx?id=70343

9. Accellera, The language for system-level modeling, design and verification. Accellera, Tech. Rep., October 2015. http://accellera.org/community/systemc/about-systemc
10. Cheung PH, Hao K, Xie F (2007) Component-based hardware/software co-simulation. In: Digital system design architectures, methods and tools. 10th Euromicro Conference on DSD 2007, August 2007, pp 265–270
11. Ren C, Huang Y, Chen H, Tian G (2014) Control software development of drive motor for electric vehicles. In: Transportation electrification Asia-Pacific (ITEC Asia-Pacific), 2014 IEEE Conference and Expo, August 2014, pp 1–6
12. Hsu Y-T, Wen Y-J, Wang S-D (2007) Embedded hardware, software design and cosimulation using user mode linux and systemc. In: Parallel processing workshops, International Conference on ICPPW 2007, September 2007, pp 17–17
13. Fitzgerald J, Pierce K, Larsen P (2014) Co-modelling and co-simulation in the engineering of systems of cyber-physical systems. In: 2014 9th International conference on system of systems engineering (SOSE), June 2014, pp 67–72
14. Brito AV, Bucher H, Oliveira H, Costa LFS, Sander O, Melcher EUK, Becker J (2015) A distributed simulation platform using HLA for complex embedded systems design. In: 2015 IEEE/ACM 19th international symposium on distributed simulation and real time applications (DS-RT), October 2015, pp 195–202
15. Lasnier G, Cardoso J, Siron P, Pagetti C, Derler P (2013) Distributed simulation of heterogeneous and real-time systems. In: 2013 IEEE/ACM 17th international symposium on distributed simulation and real time applications, October 2013, pp 55–62
16. Kanduri A, Rahmani AM, Liljeberg P, Wan K, Man KL, Plosila J (2013) A multicore approach to model-based analysis and design of cyber-physical systems. In: 2013 International SoC design conference (ISOCC), November 2013, pp 278–281
17. Brito AV, Negreiros AV, Roth C, Sander O, Becker J (2013) Development and evaluation of distributed simulation of embedded systems using ptolemy and HLA. In: Proceedings of the 2013 IEEE/ACM 17th international symposium on distributed simulation and real time applications. IEEE Computer Society, pp 189–196
18. IEEE standard for modeling and simulation (M&S) high level architecture (HLA)—federate interface specification. IEEE Std 1516.1-2010 (Revision of IEEE Std 1516.1-2000), pp 1–378, August 2010
19. IEEE standard for modeling and simulation (M&S) high level architecture (HLA)—object model template (OMT) specification. IEEE Std 1516.2-2010 (Revision of IEEE Std 1516.2-2000)
20. Negreiros ALV, Brito AV (2012) The development of a methodology with a tool support to the distributed simulation of heterogeneous and complexes embedded systems. In: Brazilian symposium on computing system engineering (SBESC), November 2012, pp 37–42