Multi-modelling of Cooperative Swarms

Georgios Zervakis[1], Ken Pierce[2], and Carl Gamble[2]

[1] Elsevier, United Kingdom
g.zervakis@elsevier.com
[2] Newcastle University, United Kingdom
{kenneth.pierce,carl.gamble}@newcastle.ac.uk

Abstract. A major challenge in multi-modelling and co-simulation of cyber-physical systems (CPSs) using distributed control, such as swarms of autonomous Unmanned Aerial Vehicles (UAVs), is the need to model distributed controller-hardware pairs where communication between controllers using complex types is required. Co-simulation standards such as the Functional Mock-up Interface (FMI) only supports simple scalar types. This makes the protocol easy to adopt for new tools, but is limiting where a richer form of data exchange is required, such as distributed controllers. This paper applies previous work on adding an explicit network VDM model, called an ether, to a multi-model by deploying it to a more complex multi-model, specifically swarm of UAVs.

Keywords: multi-modelling, swarms, FMI, co-simulation

1 Introduction

The design of cyber-physical systems (CPSs) requires engineers from across disciplines to collaborate in order build the collections of physical, control and network systems from which they are built. Swarms of Unmanned Aerial Vehicles (UAVs) represent CPSs; they require their hardware, software and elements to be closely integrated if they are to function properly. While model-based engineering approaches for CPSs are desirable, the distinct modelling formalisms used by different disciplines is a barrier to collaborative design. A promising approach to overcome this is multi-modelling, where individual models of the components, retained in their appropriate tools and formalisms, are combined into system-level models that can be analysed through co-simulation.

Multi-modelling of distributed CPSs such as UAV swarms presents a challenge to current approaches. Such swarms form sets of controller-hardware pairs within the multi-model, which must also communicate between pairs in order to model distributed control (c.f. Figure 2a). Co-simulation standards such as the Functional Mock-up Interface (FMI)[3] support simple scalar types and strings, and not the rich set of data abstractions familiar to engineers modelling software components in formalisms such as Vienna Development Method (VDM) [11]. Also as the number of units in the swarm increases, direct connection between all controller models becomes unwieldy.

[3] http://fmi-standard.org/

In previous work [5], introduction of an explicit network model to a multi-model was demonstrated and applied to a building case study. This approach overcomes both the limited number of types available and the increasing number of connections required as swarms increase inside. In this paper we present the multi-modelling of a swarm of UAVs using the INTO-CPS technologies[4]. The multi-model comprises the simple network model and a scalable set of controller-hardware model pairs, modelled in VDM and 20-sim, respectively (see Section 2). We demonstrate the potential of this approach and consider weaknesses and future directions based on experiences.

The remainder of the paper is structured as follows. Section 2 describes the modelling technologies used in the paper. Section 3 introduces the case study, namely a swarm of UAVs. Section 4 describes the modelling of the case study, focusing on the controller and network model. Section 5 shows results of co-simulation of the multi-model. Finally, Section 6 draws conclusions and presents future work.

2 Background

In this section we briefly describe the modelling tools used for the multi-modelling covered in Section 4: the INTO-CPS technologies, VDM-RT and 20-sim.

2.1 INTO-CPS Technologies

The INTO-CPS technology allows the user to build and analyse *multi-models* comprising multiple constituent models [9], using both discrete-event (DE) and continuous-time (CT) formalisms. INTO-CPS adopts the Functional Mock-up Interface (FMI) standard, where constituent models are packaged as Functional Mockup Units (FMUs). Over 30 different tools can produce FMUs, with partial or upcoming support bringing the total to over 100[5]. INTO-CPS provides a co-simulation engine, Maestro, which acts as a co-simulation master algorithm, offering both fixed step size and variable step size co-simulation. The INTO-CPS technologies are supported by the INTO-CPS Association, a group of organisations working on the technologies, based around a community of industrial users. As their organisations are part of the Association, INTO-CPS has guaranteed support for FMUs produced by Overture and 20-sim, described below.

2.2 VDM-RT

The Vienna Development Method (VDM) [11] is a well-established formal method for systematic analysis of system specifications. VDM++ is an object-oriented extension of the original VDM Specification Language (VDM-SL) used for the development of computer-based systems and software. VDM-RT is an extension of VDM++, intended for the specification of real-time embedded and distributed systems [16]; it includes features required for description of real-time controllers, in particular native support for a computational time model and distribution of functionality between compute units, interconnected by a communications network.

[4] http://into-cps.org/

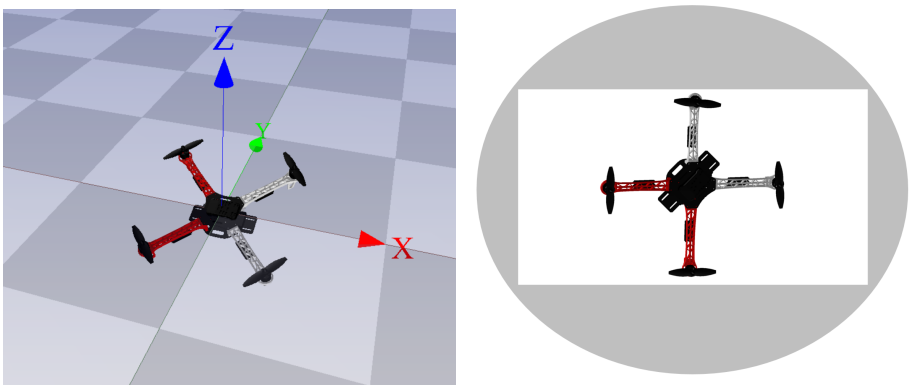
[5] http://fmi-standard.org/tools/

2.3 20-sim

20-sim[6] represents continuous-time (CT) models using graphs of connected blocks or icons [8]. Blocks and icons contain differential equations or code that represent physical phenomena, while the connections denote channels over which the phenomena interact. These channels may be one-way (signals) or two-way (bonds). The bi-directional bonds carry the domain-independent values of effort and flow— these map to familiar physical concepts, e.g. voltage and current. Bonds offer a powerful, compositional and domain-independent way to model physical phenomena. These CT models are solved numerically to yield high-fidelity simulations of physical components.

3 Case Study

Swarms of relatively inexpensive UAVs have the potential to save time and money and even lives across a variety of applications, including border patrol [12], monitoring of nuclear power plants, monitoring of forest fires [4] and search missions for finding missing persons [2]. Using automated guidance and autonomous decision making, UAVs have the potential to operate for extended periods of time without significant human supervision [6]. Consequently, they can be used for wilderness search and rescue operations that may require hundreds of hours of search at low altitude, and using flight profiles close to objects where piloted systems cannot be flown [1, 14]. Even with only simple algorithms for generating search paths, a team of UAVs is more efficient than a single UAV [15], therefore the development of robust, cooperative UAV systems will lead to improved outcomes in operations such as search and rescue.



(a) (b)

Fig. 1. Quadcopter airframe and camera view

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

In this paper we consider a case study of a UAV swarm searching a geographical location. The INTO-CPS technology was applied in the design of a UAV swarm for searching and surveillance. Specifically, the swarm was tasked with collaboratively and autonomously searching for human bodies within a specified area. Each UAV in the swarm is a quadcopter, as shown in Figure 1a, a compact drone with four fixed-pitch propeller blades and requiring a low-level loop controller for aerial stability. Searching is carried by two downward-facing cameras (not modelled), covering the visual and infrared spectrum [14]. This gives a rectangular footprint (the minimum of the two cameras) as shown in Figure 1b. To search an individual area, a UAV must visit a number of locations to guarantee visual coverage of its assigned area. As a swarm, the UAVs must divide up larger search areas to be searched by individual UAVs.

4 Multi-Modelling of the UAV Swarm

4.1 Multi-model Composition

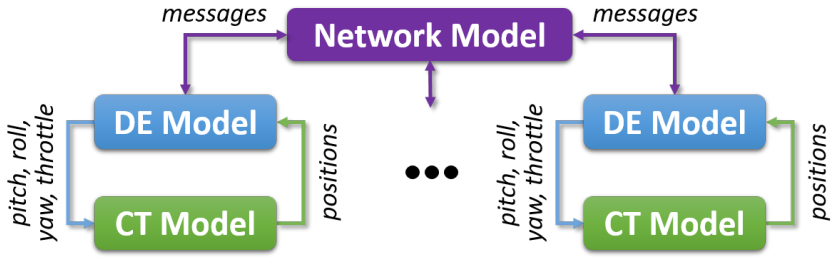
The UAV swarm is realised as a set of FMUs connected together as a multi-model. The logical connections between the DE, CT and network models are shown in Figure 2a. Each UAV is represented by a DE-CT model pair, representing the controller and hardware respectively. The DE model passes control outputs (for the motors) to the CT model, which in turn passes back positional information, i.e. x-, y- and z-coordinates. Each DE model is also connected to the network model, so that the controllers can pass messages for coordination. Figure 2b shows how these logical connections are actually realised in the FMI protocol, via the co-simulation engine (Maestro).

4.2 CT Model

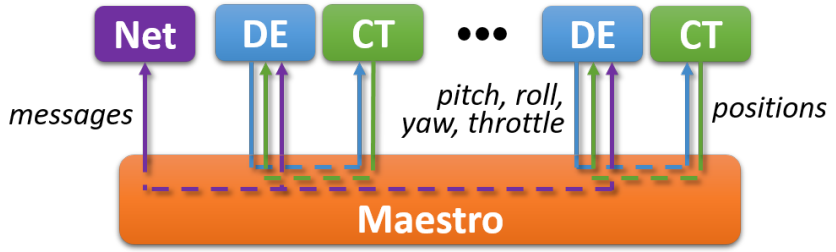
The CT model for a single UAV is shown in Figure 3. Each block is an element of the physical system (motors, rotors, frame, battery). The arrows in the top left are inputs to the model (throttle, pitch, yaw, and roll) which allows the DE model to move the UAV. The arrows on the lower half are outputs (sensor readings of position and orientation). Together these form the interface of the FMU. The physics and battery model allows the controller to be tuned accurately for minimal-energy path planning and return-to-base recharging. This model is derived from the high-fidelity model presented in [10], but incorporates loop-level control. This presents a higher-level interface to the controller and speeds up simulations as fewer synchronisation steps are needed per co-simulation.

4.3 DE Model

The DE model provides the supervisory control of the UAVs, specifically movement and distributed coordination for searching. The modes of the controller are shown in Figure 4. Each UAV searches a specific area using waypoints and visiting these in turn. The waypoints cover the entire assigned search area, including along straight line paths (see Section 5, as the UAV must pause briefly to take a clear image at each waypoint. The controller also monitors its battery usage and returns to base to recharge, resuming



(a) Logical connections between constituent models in the multi-model



(b) Connections between FMUs and the co-simulation engine

Fig. 2. Diagrams showing logical and actual multi-model compositions

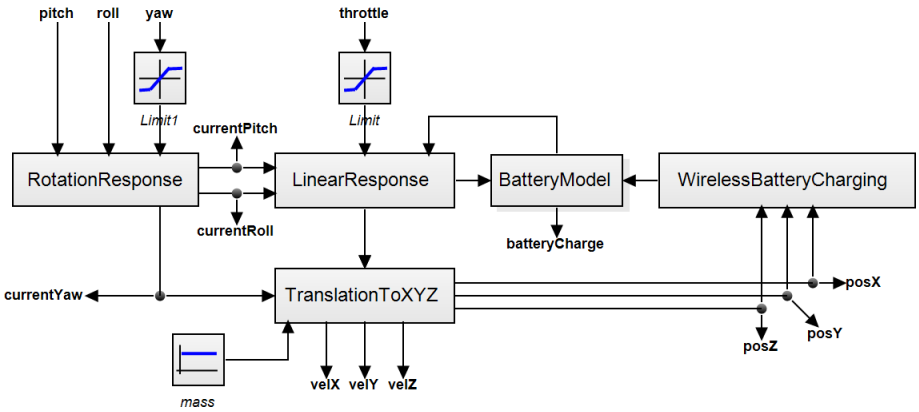


Fig. 3. Physical model of a quadcopter in 20-sim

its visiting of the waypoints until its search is complete. The costliest maneuver for a quadcopter is a U-turn [13], which the path planner takes into account by minimising U-turns when dividing and generating the assigned waypoints in the search space.

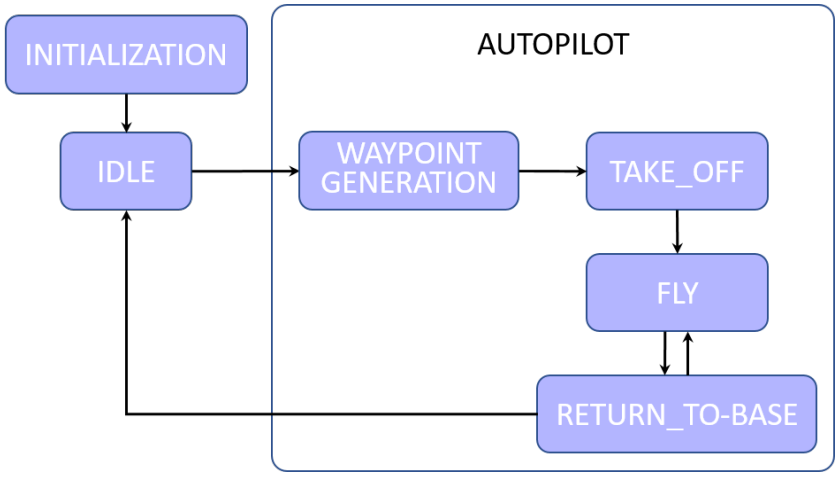


Fig. 4. Modes of the controller model

The UAV has four modes: INITIALIZATION, RETURN_TO_BASE, TAKE_OFF, and FLY. The first mode that the UAV enters is the INITIALIZATION mode, which the UAV will enter only once to generate waypoints for a specific area, and plan its trajectory, as well as to store its initial coordinates. Since the UAV starts from a base, it is essential to store those coordinates to return when it has finished its task, or when there is a need for recharging.

When the UAV has generated waypoints and trajectory and has sufficient energy, it enters the TAKE_OFF mode to carry out the mission. As we model a quadcopter able to take-off and land vertically, the TAKE_OFF mode is used to launch the UAV vertically. When the UAV reaches a desired altitude that is considered safe to start its mission, it enters the FLY mode to visit the waypoints. These can be seen in Section 5.

In FLY mode, the UAV continually determines its target position, which is the next waypoint that exists in the sequence. It is also responsible for updating the sequence with the remaining waypoints, erasing the visited waypoints. A waypoint is considered visited only if the UAV has reached approximately the position of that waypoint (in order to take an accurate photo and ensure coverage). In this model, a waypoint was considered visited if the UAV inclines less than 0.3 distance units (metres) per each coordinate.

The RETURN_TO_BASE mode forces the UAV to return to its base. The UAV enters that mode for two reasons. Firstly, the UAV enters that mode when it finishes its mission in order to return back to its base. Secondly, it enters that mode if there is insufficient energy to carry on the mission. Note the UAV does not attempt to visit extra waypoints on the way back to recharge, but this could be implemented as an

efficiency saving. In each iteration, the UAV calculates its average consumption based on the distance travelled and the battery consumption. Afterwards, it checks whether the distance from its base is greater than the distance that can be travelled, taking into account the remaining energy. A safety buffer is included, so that a UAV will return to base before its energy is exhausted, ensuring there is sufficient energy to return to base.

The actual flight controller model for each UAV is split across both the DE and CT models. The DE model for each UAV is responsible for determining the difference between the current position of the UAV and its next waypoint. This difference, along with the current velocity in the X,Y, and Z axes, become the input for three PID controllers, one each for the UAVs X axis and Y axis, that output pitch and roll angles for the drone to adopt and a third for the Z axis that outputs a throttle setting to maintain the target altitude. These pitch, roll and throttle values are sent from the DE controller to the CT model of the UAV (Figure 2a), this is consistent with an 'attitude mode' of flight control that is found on many multi-rotor UAVs. The CT model receives the pitch and roll and uses a PID controller in the 'Rotation Response' block (Figure 3) to determine the actual orientation of the UAV at any point in time. The orientation is combined with the throttle value in the 'Linear Response' block to compute the velocity of the UAV relative to its own axis, which is translated into a global velocity in the 'Translation to XYZ' block. All sensing of the UAVs orientation, position and velocity is performed in the CT model (taken directly from the computed values), with the X,Y,Z position and speeds sent back to the DE controller in the 'poisitions' message (Figure 2a).

4.4 Network Model

The network model is a single FMU that represents an abstract communications medium, called the ether [7]. We adopt this approach because connecting each controller model directly to every other controller is unwieldy, and FMI currently lacks native support for such network connections. The ether is aware of all controllers connected to it, and passes all messages received to all other UAVs. By encoding messages as strings, we can also overcome the limited types supported by FMI connections. In this multi-model, identifiers and message recipients are handled by the VDM model directly, because this would form part of the software as deployed on the real UAVs, however message handling could be added to the ether model if this was appropriate.

The network FMU used in this case study is an initial, abstract network model built in VDM as a proof of concept for modelling communications within the limitations of FMI. The model does not currently cover advanced features such as specific protocols, error handling, or data rates. The network model is also unaware of the physical world meaning that communications are not affected by distance, orientation or obstruction and so, without explicit modelling of faults elsewhere in the multi-model, the communication network is fully connected at all times.

4.5 Distributed Coordination with Communication

During INITIALIZATION, each UAV announces itself and a LEADER is selected, with the others becoming WORKER UAVs. In the current model, the selection is deterministic (the UAV with the lowest id becomes leader), however Bryans et al. [3] demonstrate a

robust, distributed election scheme that could be implemented in the final system. The LEADER then divides the search space between the available UAVs and assigns them a sub-area. Each UAV then searches its sub-area, managing waypoints and battery levels.

Using the ether FMU, each UAV is able to broadcast its id number, position, battery life, a tag id, four real numbers, an acknowledgment, a map consisting of the id numbers of the UAVs undertaking a task and the coordinates of their task, and a map consisting of the UAVs that have finished their tasks and the coordinates of their tasks. The tag id indicates which UAV should undertake a task; the four (real) numbers form two pairs of coordinates indicating the sub-area that the worker UAV should cover. The worker UAV knows when a task is intended for itself when the tag id is equal to its id number.

The acknowledgment is used from the worker UAVs to send an acknowledgment to the LEADER that they received their task. After the completion of their task, the workers send the LEADER another acknowledgment indicating that they finished it. If the LEADER receives such an acknowledgment, it erases the worker UAV from the list of the UAVs undertaking a task, and stores the UAV and its task in the map of UAVs that have finished their tasks. The LEADER does the same when it finishes its task.

The two maps are being sent from the LEADER UAV to the other members of the team, allowing them to know which UAVs have been assigned a task, and which UAVs have finished their tasks and the areas covered so far. At this stage, these maps are not used for anything, but have potential for future work to incorporate more dynamic cooperation, such as in-flight reassignment or responses to potential results.

After INITIALIZATION and TAKE_OFF, the UAV goes into FLY mode. During FLY mode, each UAV transmits its id number and position, allowing every member of the swarm to determine the other UAVs that are taking part in the mission and their position. Processing of these messages is shown in the extract in Listing 1.1. In the message tuple, the first (natural) number is the tag id, the next three (real) numbers are the x-, y-, and z-positions of the UAV, and the final (real) number is the battery level.

If a UAV is not heard from in a certain amount of time, its sub-area is reassigned to the first available UAV to finish its initial sub-area. If communication with the LEADER is lost, a new leader is selected (the UAV with the next lowest id). In this way, the swarm is resilient to lost UAVs.

5 Results

The multi-model was successfully able to simulate a range of scenarios and demonstrate the possibility of studying distributed communications and swarm behaviours with multi-modelling techniques. In these results the swarm is homogeneous, with all UAVs beginning at the same time, from the same spot, and with the same initial fuel.

A live plot of a single UAV searching an area is shown in Figure 5. Here the vertical take-off is shown as `{uav}.uav.posZ` (green), with a zig-zag searching pattern shown by `{uav}.uav.posX` and `{uav}.uav.posY` (blue and orange respectively). At 400 seconds the UAV returns to base for recharging before completing its search.

Such visualisations give intuitive feedback to software engineers about the effects of their design choices, however to better demonstrate the behaviour of the swarms, the co-simulation outputs (in CSV format) were post-processed in Matlab. A plan view of

```
public uavPosition :: x : real
                      y : real
                      z : real
                    bat : real;
...

Step() == cycles(2)
(
  -- broadcast own position
  etherOut.setValue(
    VDMUtil`val2seq_of_char[nat*real*real*real*real](
      mk_(id, posX.GetValue(), posY.GetValue(),
          posZ.GetValue(), battery.GetValue())
    )
  );

  -- receive messages
  if len etherIn.getValue() >= 2 then (
    let mk_(list,l) =
     VDMUtilDebug`seq_of_char2val[
     seq of(nat*real*real*real*real)](etherIn.getValue())
     in if list then (
       for all z in set inds l do (
         let x = l(z) in (uavPositions:= uavPositions ++
           {x.#1 |-> mk_uavPosition(x.#2,x.#3,x.#4,x.#5)};
         )
       )
     )
  )
)
```

Listing 1.1. Message processing in the UAV controller VDM model

the data shown in Figure 5 is given in Figure 6a. Here the waypoints are marked as triangles, with the path of the UAV as a black line, including its return to base to recharge. The UAV controller divided the space along the long axis to minimise U-turns.

Figure 6c shows a UAV swarm dividing and searching an area cooperatively. As in Figure 6a this is a plan view of the x- and y-position of the UAVs. In this co-simulation, there are three UAVs that can potentially join the swarm, however as the area is smaller than in Figure 6a, negotiations result in two of the three UAVs take a divide-and-conquer approach to perform the search cooperatively (represented by the black and green lines). Note that they fly in a latitude-first direction to minimise U-turns. Figure 6b shows how the camera footprint captures the searched area.

Figure 6d shows the same three-UAV swarm searching another area. A communication fault is included in the UAV controller FMU and when triggered, by a combination of time and UAV id, it blocks that UAV from communicating with the remainder of the

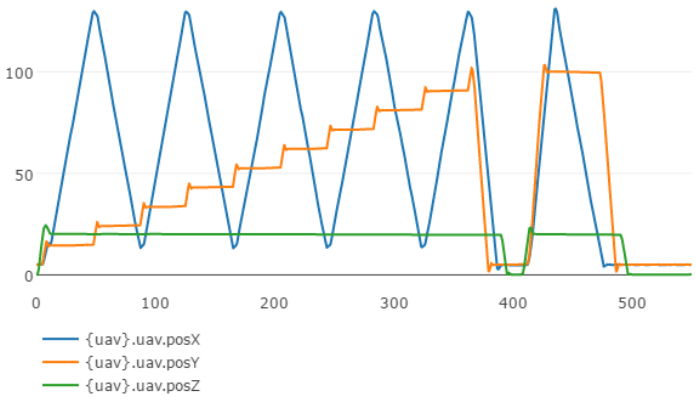


Fig. 5. Live plot of the position of a single UAV during a search

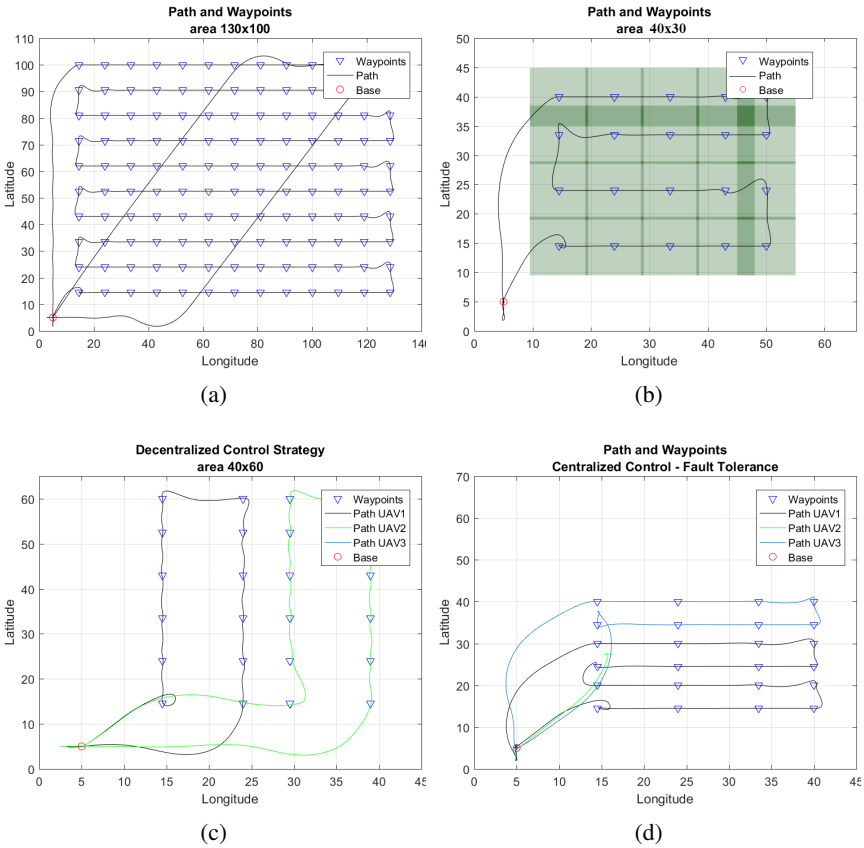


Fig. 6. Plan view of UAV searches

swarm. The second UAV (green line) is then 'lost' from the swarm, as can be seen by the line ending at around point (17,28). After a timeout, the leader decides that this UAV is missing as no further communications were received. The leader then reassigns this area to another UAV (specifically itself, in this scenario), which completes searching this area after completing its own sub-area.

6 Conclusions and Future Work

In this paper we applied a multi-modelling approach to a swarm of UAVs. The multi-model uses a simple network model [5] to allow a set of UAV controllers to communicate using complex data types in order to model realistic communication protocols. Each controller model is paired with a high-fidelity physics model. The results presented demonstrate the potential of this paradigm for multi-modelling of swarms and other CPSs with distributed control within the constraints of the FMI standard.

The network FMU used in this case study is an initial, abstract network model built in VDM as a proof of concept for modelling communications within the limitations of FMI. As highlighted in previous work [5], there are some drawbacks to that specific network FMU approach. Firstly, messages require a number of co-simulation cycles to reach recipients, so careful consideration of co-simulation synchronisation timing is required[7]. Secondly, the model does not currently cover advanced features such as specific protocols, error handling, or data rates. The openness of FMI means that network models can be swapped, either with an improved version of the existing FMU or replacing it with a dedicated network model such as OpNet or NS2, which in combination with an increase in multi-model synchronisation rate to some multiple of the controller frequency, could improve network modelling fidelity.

However, increasing the synchronisation rate of whole multi-model is undesirable since synchronisation additional synchronisation steps above the frequency of a DE controller have no effect on behaviour but will increase the time to complete a simulation. The multi-model could be altered to make use of the hierarchical co-simulation feature of Maestro to target increasing the frequency of the controller-ether synchronisations while leaving the controller-uav physics synchronisations at their current rate.

Improvement of the network model is a key next step. One such improvement would be the addition of fault behaviour to better support the behaviour that is demonstrated in Figure 6d. The goal here would be to move the fault triggering from its current hard coded state to something that may be altered more easily, such as either parameters of the FMU or perhaps a script that could be read in at the beginning of a simulation.

Another intriguing next step forward is to observe in Figure 2a the potential for an analogue of the network model to link together CT models on the bottom of the diagram. This would suggest an "environmental ether" model representing physical interactions between components. This could include, for example, physical obstacles, occlusion of line-of-site for communications, and even collisions. Physical interactions such as contact and collision models are particularly challenging as they require tightly-coupled interactions.

[7] The Maestro co-simulation engine supports minimum frequency constraints which could alleviate this problem somewhat

Acknowledgements

The work reported here is supported by the EU Horizon 2020 Projects "Integrated Tool Chain for Model-based Design of Cyber-Physical Systems" (INTO-CPS, Grant Agreement 644047) and "CPS Engineering Labs - expediting and accelerating the realization of cyber-physical systems" (CPSELabs, Grant Agreement 644400).

References

1. Adams, J.A., Humphrey, C.M., Goodrich, M.A., Cooper, J.L., Morse, B.S., Engh, C., Rasmussen, N.: Cognitive task analysis for developing unmanned aerial vehicle wilderness search support. Journal of Cognitive Engineering and Decision Making 3(1), 1–26 (2009), https://doi.org/10.1518/155534309X431926
2. Beard, R.W., McLain, T.W., Nelson, D.B., Kingston, D., Johanson, D.: Decentralized cooperative aerial surveillance using fixed-wing miniature uavs. Proceedings of the IEEE 94(7), 1306–1324 (July 2006)
3. Bryans, J., Fitzgerald, J., Payne, R., Kristensen, K.: Maintaining Emergence in Systems of Systems Integration: a Contractual Approach using SysML. In: INCOSE International Symposium on System Engineering. INCOSE (2014)
4. Casbeer, D.W., Kingston, D.B., Beard, R.W., McLain, T.W.: Cooperative forest fire surveillance using a team of small unmanned air vehicles. International Journal of Systems Science 37(6), 351–360 (2006), https://doi.org/10.1080/00207720500438480
5. Couto, L.D., Pierce, K.: Modelling Network Connections in FMI with an Explicit Network Model. In: J. S. Fitzgerald, P.W.V.T.J., Oda, T. (eds.) The 15th Overture Workshop. pp. 31–43. Newcastle University, Computing Science. Technical Report Series. CS-TR- 1513, Newcastle, England (September 2017)
6. DeLima, P., Pack, D.: Toward developing an optimal cooperative search algorithm for multiple unmanned aerial vehicles. In: 2008 International Symposium on Collaborative Technologies and Systems. pp. 506–512 (May 2008)
7. Fitzgerald, J., Larsen, P.G., Verhoef, M. (eds.): Collaborative Design for Embedded Systems – Co-modelling and Co-simulation. Springer (2014), http://link.springer.com/book/10.1007/978-3-642-54118-6
8. Kleijn, C.: Modelling and Simulation of Fluid Power Systems with 20-sim. Intl. Journal of Fluid Power 7(3) (November 2006)
9. Larsen, P.G., Fitzgerald, J., Woodcock, J., Fritzson, P., Brauer, J., Kleijn, C., Lecomte, T., Pfeil, M., Green, O., Basagiannis, S., Sadovykh, A.: Integrated tool chain for model-based design of Cyber-Physical Systems: The INTO-CPS project. In: 2016 2nd International Workshop on Modelling, Analysis, and Control of Complex CPS (CPS Data). IEEE, Vienna, Austria (April 2016), http://ieeexplore.ieee.org/document/7496424/
10. Larsen, P.G., Fitzgerald, J., Woodcock, J., Nilsson, R., Gamble, C., Foster, S.: Towards semantically integrated models and tools for cyber-physical systems design. In: Margaria, T., Steffen, B. (eds.) Leveraging Applications of Formal Methods, Verification and Validation, Proc 7th Intl. Symp. Lecture Notes in Computer Science, vol. 9953, pp. 171–186. Springer International Publishing (2016)
11. Larsen, P.G., Lausdahl, K., Battle, N., Fitzgerald, J., Wolff, S., Sahara, S., Verhoef, M., Tran-Jørgensen, P.W.V., Oda, T.: VDM-10 Language Manual. Tech. Rep. TR-001, The Overture Initiative, www.overturetool.org (April 2013)
12. Leng, G., Qian, Z., Govindaraju, V.: Multi-UAV surveillance over forested regions. Photogrammetric Engineering & Remote Sensing 80(12) (2014)

13. Maza, I., Ollero, A.: Multiple uav cooperative searching operation using polygon area decomposition and efficient coverage algorithms. In: Alami, R., Chatila, R., Asama, H. (eds.) Distributed Autonomous Robotic Systems 6. pp. 221–230. Springer Japan, Tokyo (2007)
14. Rudol, P., Doherty, P.: Human body detection and geolocalization for uav search and rescue missions using color and thermal imagery. In: 2008 IEEE Aerospace Conference. pp. 1–8 (March 2008)
15. Sujit, P.B., Ghose, D.: Search using multiple uavs with flight time constraints. IEEE Transactions on Aerospace and Electronic Systems 40(2), 491–509 (April 2004)
16. Verhoef, M., Larsen, P.G.: Enhancing VDM++ for Modeling Distributed Embedded Real-time Systems. Tech. rep., Radboud University Nijmegen (March 2006), a preliminary version of this report is available on-line at `http://www.cs.ru.nl/~marcelv/vdm/`