Proposed options for simulator architecture

A best-effort comparison for specific use case

Marek S. Łukasiewicz

Polimi DAER

February 6, 2024

Section 1

Goal definition

Problem statement

Integrate available hardware and software resources to provide flight simulation for the following projects:

- Rotorcraft-Pilot Coupling
- Manned-UnManned Teaming
- G-Seat motion cueing
- *Probable future research, not yet defined*

Methodology

The hardware components are mostly done, so for remaining work software development techniques will be used.

Behavior-driven development will be employed to guide the technical requirements based on the needs of the users. Using plain prose instead of technical specification in the early phase is hoped to involve more stakeholders in the process.

The example needs have been written as user stories, following the format:

As a *(role)* ***I can*** *(capability),* ***so that*** *(receive a benefit)*

User stories — high level goals

As a researcher **I can** add new flight models, measurement devices and cueing systems, **so that** the simulator is useful for my research.

As a project leader **I can** use the simulator for commercial purposes without paying for any licenses, **so that** I can involve collaborators from industry environment.

As a project leader **I can** use the simulator without relying on any external service **so that** I am sure the simulator will work in the future, regardless of an external company

As a professor **I can** easily introduce students to the facility, **so that** they do practical projects.

As a student **I can** use widespread solutions, standards and libraries, **so that** I get practical experience for my career after graduating.

User stories — conducting studies

As a publication or thesis author **I can** easily gather all trial data into a single entity, **so that** they can be analyzed and presented in a written work.

As a user conducting trials **I can** control the whole simulator on my own using a single application, **so that** there are less people to schedule for a trial with a test subject, and I can iterate on my own.

As a human factors researcher **I can** see simulated view with an imperceptible delay, **so that** a human-in-the-loop piloting is viable.

As a human factors researcher **I can** use the motion platform in closed loop mode, **so that** the simulation realism for the pilot is increased.

User stories — compatibility

As a RPC project participant **I can** use Simulink and MBDyn models, **so that** I can reuse work already done in the project.

As a human factors researcher **I can** connect the simulation infrastructure to other flight simulation software eg. FlightGear or X-Plane, **so that** off-the-shelf visual models of cockpit and aircraft can be used.

As a user working on UAV support for HEMS missions (eg. MUM-T) **I can** communicate multiple aircraft (including unmanned) simultaneously, **so that** I can run shared simulations.

As a user of UAVs **I can** connect PX4 simulation, **so that** I can collaborate with the drone lab and industrial partners.

As a user of G-Seat **I can** connect the same simulator to moving platform and other cueing devices, **so that** a comparative study can be performed

User stories — developer experience

As a simulator developer **I can** reuse common elements in different configurations, **so that** there is less work repeated to prepare the simulator for a new study.

As a user adding a new device or flight model **I can** read a well-written and detailed documentation, **so that** the development process is feasible.

Requirements summary

Some requirements have been identified as a must, and any solution not satisfying them has been excluded from the comparison:

- Modular architecture, allowing for separate programs, written in different programmming languages, running on different computers to participate in the simulation
  - Including cooperation when run under Windows and Linux
- Reference implementations and documentation available online for free
  - Found for Python, C and C++; some for MATLAB are paid only
- Ability to monitor all exchanged information, for purpose of experimental data collection and debugging

Differentiating features

- **DEF: Relevant definitions** — Does it come with definitions for information we want to communicate?
  - Information about aircraft like position, attitude etc. in physical units
  - Concept of subsystems of an aircraft
- **EZ: Ease of use** — Does it have good documentation? How can it be used with MATLAB? *(all of them have C, C++ and Python support)*
- **SC: Software constraints** — Does it require a redesign of existing application? Does it impose an execution model like some dynamical co-simulation standards?
- **APP: Wider applicability** — Is it used in the industry? Is it a useful skill to have?

Non-goals

The system need not be decentralized. In practice, the simulator operator actually *wants* a central control panel to manage the simulation. The low risk of simulation compared to regular operations, allows a single point of failure in exchange for a simpler architecture and usage.

Section 2

Proposed networking solutions

Overview

The options to proceed have been grouped in three main ways:

1. Fully custom software as needed
2. Following an aerospace standard
3. Mixed approach using IT standards

Approach evaluation

Some difficulties about the evaluation:

- A third-party comparison in an academia context would be preferred. Some were found, but for a more narrow context
- Documents about a given solution are typically written by authors, and understate its disadvantages
- Industry examples are very rarely shared publicly, or when they are they have the purpose of advertising a solution (see above)
- Experience in the industry has to be stripped of context to avoid breaching the non-disclosure agreement

Select approaches

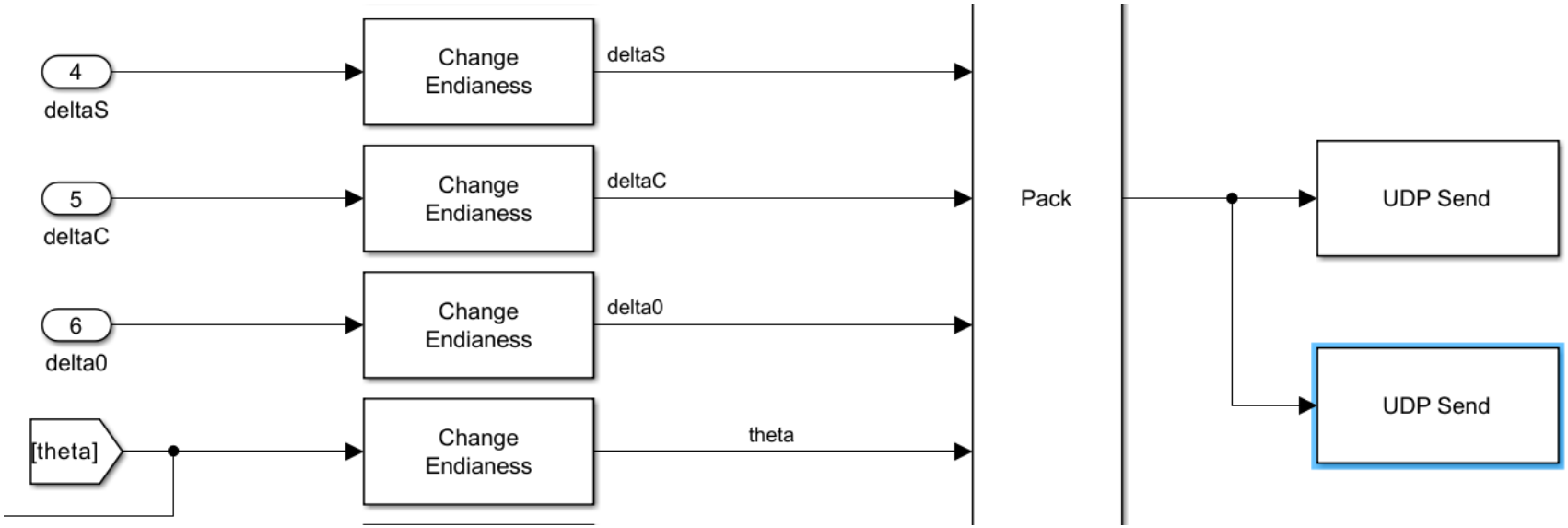
Most promising options chosen from a wider set (full table).

1. Custom software - status quo
2. Aerospace standards
   - HLA
   - DIS
   - DDS
   - MAVLink
3. Mixed approach with IT standards
   - Definitions and serialisation
   - Module discovery and management
   - Transport protocol

Status quo

Raw bytes over UDP from every part of simulation to any other component that needs it. Needs manual configuration of ports and addresses. When a message changes, silently fails or worse, gets corrupted data.

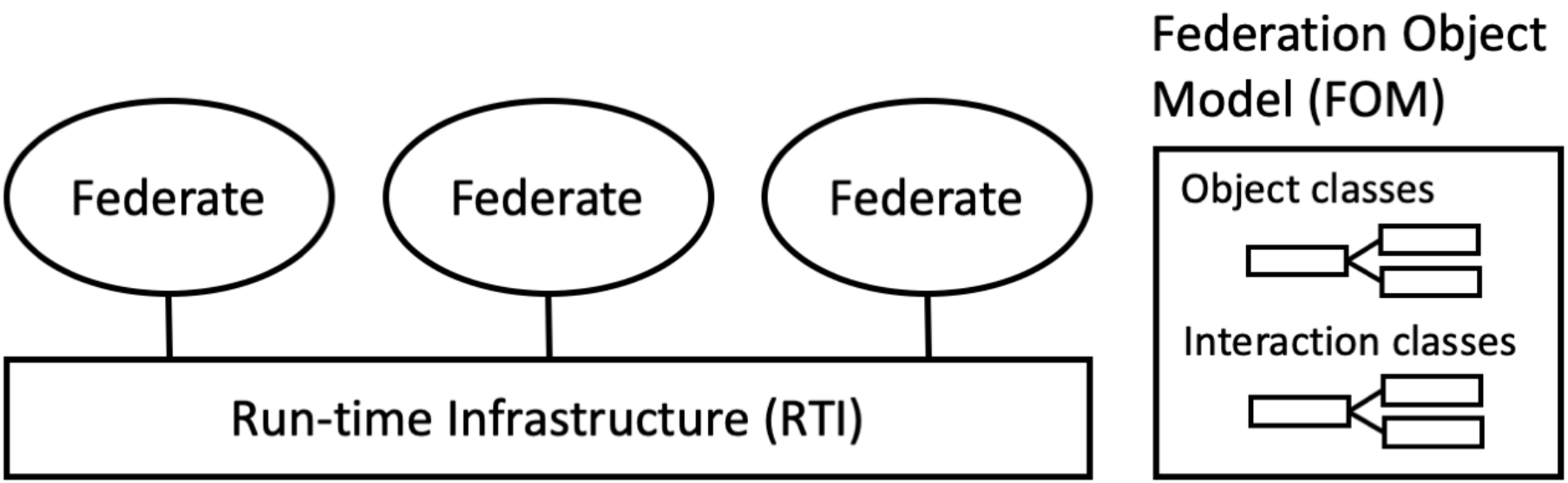
- DEF: bad — ad hoc definitions with little planning
- EZ: neutral – used to be simple for two modules, but big problems now
- SC: good — just add raw UDP output
- APP: very bad — teaching bad practice, compatible with nothing



High-Level Architecture (HLA)

Designed as the ultimate universal simulation standard, but standardised C++/Java functions instead of data actually sent on the network. This means that every component needs to link the implementation from the same vendor.

This design makes sense for huge scale simulations of hundreds of objects, but is too much for users seeking only to connect modules over network.



HLA Architecture by WikiMedia user 61cc

High-Level Architecture cont.

- DEF: very good
  - there are FOMs meant specifically for simulation
  - it is also expected to handle subsystems
- EZ: neutral
  - there are GUI editors, and fairly good tutorials
  - need to dynamically link an RTI; either adapt to MATLAB ourselves, or a paid plugin
- SC: bad — requires a Federate to be controlled by the bus, allowing start/stop, lockstep etc. but simulation needs to be written around HLA callbacks
- APP: neutral
  - recommended by NATO, in practice not as popular as expected
  - many tools are paid

Distributed Interactive Simulation (DIS)

Simple protocol to broadcast events in a wargame developed for DARPA in 1980s. HLA was designed to address disadvantages of DIS, now both are worked on by SISO

- DEF: bad — set of predefined messages (PDUs) relevant for wargames, but it is fixed and missing subsystems
- EZ: neutral – not complicated but few resources; either adapt to MATLAB ourselves, or a paid plugin
- SC: good – defines a set of relevant messages to be broadcasted
- APP: neutral
  - in theory superseded by HLA, officially cancelled from NATO in 2010
  - still being mentioned in literature as late as 2022, hinting at widespread use (*backed by anecdotal evidence*)

Data Distribution Service (DDS)

Shares data between nodes through a publish-subscribe system where values can be written to and read from named topics. Has a decentralised, peer-to-peer architecture with automatic discovery. Every implementation should be compatible.

- DEF: neutral — structured way to define our data through topics, but no common set for flight simulation (maybe AEON by NASA, but not accessible online)
- EZ: good — abstracts away all network, DDS Blockset for Simulink;
  - tried using it, but only free Python library (except ROS) unfinished and abandoned
- SC: neutral — data seems to be "just available", but requires some setup of threads and the DDS library
- APP: good — selected as basis for Robot Operating System (ROS)

MAVLink

Developed for communicating with Unmanned Aerial Vehicles (a.k.a. drones), and onboard drone components. Is suitable for resource-constrained systems. In practice, successfully ensures compatibility between different autopilots, gimbals and ground control stations.

Has extensive online documentation with sub-protocols built on top of it for features like node discovery, setting parameters, reliable commands, mission profiles (sequence of commands with some conditions) and else.

Beware of exposition bias, I have been using this since 2016

MAVLink cont.

- DEF: good
  - common set of definitions for basic UAV information, which would be applicable to any aircraft
  - built to be extended by users, can contribute back our dialect for standardisation with anyone else using it for a simulator
- EZ: good — great documentation, multiple tutorials, chat with open source community, included in MATLAB UAV Toolbox
- SC: good — just defines messages that could be sent over any transport
- APP: good
  - already used by colleagues at DAER and industrial partners who work with UAVs (both use PX4)
  - MAVLink users have already developed adapters to other software like FlightGear
  - current efforts to make it a US Gov. standard for UAV interoperation

Mixed approach with IT standards

With this approach, the simulator would be connected using technologies popular in mainstream software engineering, which would give up compatibility with other flight simulation specifically. But possibly make it easier to program modules, using ubiquitous tools like the web browser and human-readable formats.

Another disadvantage is that with less constraints there are many more decisions and design work to be made, on various levels:

- Defining all names and units for our data, preferably in a structured way (OpenAPI? AsyncAPI?)
- Representation for transferring it over network (JSON? MsgPack? Capnproto?)
- Discovering different nodes and/or broadcasting (ZeroMQ? MQTT? HTTP and WebSockets?)

I am also biased by experience working with these, but a chance for prior exposition is a key advantage of this approach

Mixed approach cont.

- DEF: neutral — no standard for aircraft; could mimic vocabulary from some other application, but still left to be designed
- EZ: neutral — best resources for the underlying tools between the options, but there would be a need to extensively document how the conventions are applied specifically here; some are built-in in MATLAB
- SC: good — the solution would be custom-designed for this specific purpose
- APP: neutral — knowledge and skills learned using this custom-made solution would only be applicable to this simulator, or general-purpose web development, not aerospace

Section 3

Conclusion

Evaluation summary

Solution	Relevant Definitions	Ease of use	Software Constraints	Wider applicability
Current	bad	neutral	good	very bad
HLA	very good	neutral	bad	neutral
DIS	bad	neutral	good	neutral
DDS	neutral	good	neutral	good
MAVLink	good	good	good	good
Mixed	neutral	neutral	good	neutral

Author's preference

From this documentation analysis not backed by prototyping, it seems that **MAVLink** is the preferred choice to satisfy the unique requirements between research and teaching.

The obvious critique is that the protocol has not been designed specifically for crewed aircraft simulation. Some extensions will be required, for example for controlling the moving platform. But for such specific devices, none of other solutions already define it, and MAVLink is among these with clear extension mechanism.

Proposed implementation details

- Develop a central application **with graphical user interface** (GUI) to act as message router, publish-subscribe broker, status viewer, configuration interface and experiment data logger
  - Every module would only communicate with this application
- Wherever possible, reuse existing messages, sub-protocols and other conventions from the common dialect
- When a custom functionality is needed, extend the protocol with appropriate definitions *instead of misusing existing ones*
- Publish both the **subset of common messages actually used** and our extensions as a documentation page identical to the common documentation, but limited only to relevant content

Contact

I invite comments on the topic, your feedback will be most appreciated.

- marek.lukasiewicz@polimi.it
- B14, room 011
- Comment in the table with all options on Google Drive, specific references are also linked there.