AirborneCPS: A Simulator for Functional Dependencies in Cyber Physical Systems

A Traffic Collision Avoidance System Implementation

William Cook, Andrew Driscoll, Bastian Tenbergen*
Department of Computer Science
State University of New York at Oswego
Oswego, NY, USA
wcook3@oswego.edu, adriscol@oswego.edu, bastian.tenbergen@oswego.edu*
* corresponding author

Abstract—**The term "Cyber Physical System" (CPS) has been used in the recent years to describe a system type, which makes use of powerful communication networks to functionally combine systems that were previously thought of as independent. The common theme of CPSs is that through communication, CPS can make decisions together and achieve common goals. In contrast to traditional system types such as embedded systems, CPS hence are functionally dependent on one another. Yet, their functional dependence may cause unforeseen runtime behavior, e.g., when a CPS becomes unavailable, but others depend on its correct operation. During development of any individual CPS, this runtime behavior must hence be predicted, and the system must be developed with the appropriate level of robustness. Since at present, research is mainly concerned with the impact of functional dependence in CPS on development, the impact on runtime behavior is mere conjecture. In this paper, we present AirborneCPS, a simulation tool for functionally dependent Cyber Physical Systems to allow simulating runtime behavior.**

Index Terms—**Cyber Physical System, Traffic Collision Avoidance System, Simulator, Requirements Engineering, Implementation, Functional Dependencies, Runtime Behavior**

I. Introduction

For the past several years, Cyber Physical Systems (CPS) have emerged as a new system type like embedded systems or information systems [1], [2]. CPS are highly context-dependent, observe the world through sensors, act upon it through actuators, and communicate with one another through powerful communication networks [3]. What distinguishes CPS from traditional systems is the type of collaboration that occurs during runtime: while embedded systems also communicate with other embedded systems, this functional dependency is limited to what the embedded system has been designed to do. For example, a brake control unit in a car is typically designed by some supplier according to a specification provided by the car's manufacturer (or at least in conjunction with the needs of the manufacturer). In consequence, albeit the supplier likely aims to sell their brake control unit to competing manufacturers with different needs and specifications, at the time the brake control unit is implemented and deployed into a specific car model, the functional dependencies between the brake control unit and other systems is fixed [4]. In other words, it is known what communication bus will be used, which other systems are involved in the functional interaction, what interfaces exist, and what information is exchanged between them.

In CPS, however, these assumptions (at design time) can only be relied on with regard to the immediate interactions within the same super-system. For example, albeit the brake control unit has been designed to work with a specific communication bus within the car, a cyber physical brake control unit could conceivably also communicate the brake intent, along with the car's estimated negative acceleration to other cyber physical brake control units in surrounding cars. The purpose of such a functional collaboration could be to warn following vehicles of a rapid deceleration, hence warn drivers in those vehicles preemptively and possibly apply the brakes, too, in order to avoid a rear-end collision. Yet, during development of such a cyber physical brake control unit, assumptions about the runtime interactions with other brake control units are unreliable. For example, it cannot be known at design time [5]:

- which specific type of brake control unit a following vehicle is equipped with;
- if that brake control unit is cyber physical (or possibly some legacy system);
- if the communication protocol is compatible; or
- if the driver choses to ignore possible warnings issued by the brake control unit.

Albeit standardization helps in this matter to some extent, the problem unreliable functional dependence remains. CPS therefore pose new challenges for the development process [6] due to open context [4], [5], due to runtime adaptiveness [7], [8], or the need to interact with humans [9], which is particularly hard to predict at design time [4, p.]. One way to investigate runtime interactions of CPS is to simulate their runtime behavior.

In this paper, we present AirborneCPS: an implementation of a avionic Traffic Collision Avoidance System (TCAS, see [10]) to simulate functional dependencies in CPS. The remainder of this paper is structured as follows. Section II discusses the related work on functional dependencies in CPS as well as their simulation. Section IV gives a brief overview of TCAS and outlines why it is a suitable surrogate to functional dependencies in CPS. Section IV gives an overview over the feature set and implementation of the simulation tool AirborneCPS before Section V gives an outlook on future directions of its implementation.

II. Functional Dependencies in CPS

Defining the term "Cyber Physical Systems" is exceedingly difficult, due to the myriad of competing interests and aspects of researchers working in the field. Instead of a formal definition for CPS often their properties are defined (see, e.g., [8], [11], [12]). CPS share properties with IoT-devices (e.g., sensing the environment and transmitting sensor data to other IoT-devices, see [13]) and components in systems-of-systems (e.g., functional collaboration of systems of systems, see [14]). However, what sets CPS apart from IoT-devices and systems-of-systems is the type of functional dependencies that can occur at runtime. Since CPS have a physical manifestation in the world, their operational context (i.e., the external systems and human users CPS interact with), can change dynamically. Consider the example of the automotive cyber physical brake control unit from Section I: at any given moment during operation, a car could drive out of communication range, hence terminating the functional dependency between two (or more) CPS. Based on [3], [15], the following types of functional dependencies can be established:

- ***homogeneous & static:*** every CPS is of the same make and model as every other CPS it interacts with, and the number of interacting CPS does not change. This is the case in, e.g., distributed energy prosumer architectures, see [16].
- ***homogenous & dynamic:*** every CPS is the of the same make and model as every other CPS, but the number of interacting CPS can change dynamically at runtime. For example, this is the case in automated traffic regulation [17] or smart city infrastructures [18].
- ***heterogeneous & static:*** every CPS interacts with CPS of different makes, models, or revisions, and possibly with other non-cyber physical systems, but the number of interacting systems does not change. This is typical for IoT-based systems, e.g., in the automation domain, where IoT-sensors assist in regulating motors, pumps, etc., see [19].
- ***heterogenous & dynamic:*** every CPS interacts with CPS of different makes, models, revisions, or non-CPS, and the number of involved interacting system changes at runtime. Examples include the example from Section I, but also cyber physical adaptive cruise control units (e.g., [4], [5]).

As outlined in Section I, the type of functional dependency along with the fact that at runtime, functional dependencies can change, poses considerable challenges for developing individual CPSs and their interaction with other CPSs. Simulation of runtime behavior is a sensible remedy, and in the area of CPS, some simulation approaches and tools have been proposed (e.g., [16], [20], [21]). However, these simulators often focus on energy and power systems (e.g., [16], [22], [23]) and are hence limited to simulating homogeneous and static functional dependencies. A notable exception is the co-simulation platform proposed by Al-Hammouri et al. [24], [25] and the "HybridSim" toolchain [26]. These approaches are in principle application to all of the above mentioned functional dependency types. Yet, both HybridSim as well as Al-Hammouri et al.'s approach focus on component simulation and observation of communication behavior, rather than on the functional collaboration as outlined above.

III. Simulation of Functional Dependencies in CPS

To focus simulation on runtime interaction effects between several CPS, we have selected an airborne Traffic Collision Avoidance System (TCAS, see [10]) as a surrogate representative of functionally dependent CPS. A TCAS is a pilot assistance system aboard modern aircraft, which alerts the crew to other nearby aircraft inside the airspace head (i.e. "protection volume") on a collision course (i.e. "traffic advisory, " TA), and if necessary, instructs pilots to climb and/or descent to increase the separation altitude to the other aircraft ("resolution advisory," RA). We have selected a TCAS (as opposed to, e.g., the cyber physical brake control unit from Section I) for three reasons:

1. The very nature of airborne collision avoidance implies that TCAS interact with one another, if there are other aircraft close by. For example, TCAS are statically functionally dependent, if there is a known set of multiple TCAS instances in the vicinity of the own aircraft all aircraft must be coordinated such that no collision occurs, and dynamically functionally dependent in all other cases. The columns in Table I **Error! Reference source not found.**
2. As there are different aircraft types, the types of TCAS systems also differ. Hence, homogeneous and heterogeneous functional dependencies can be simulated. Example scenarios are provided in the rows of Table I.
3. In contrast to automotive domain, where simulators are available, but typically proprietary, limited in realism, or difficult to extend, in the avionics domain, the need for meticulous pilot training and the effort invested by aviation enthusiasts has resulted in the commercial availability of highly realistic and easy-to-extend simulation platforms, which are readily available.

TABLE I. Types of Functional Dependencies in TCAS Operation

	Dynamic	**Static**
Homogeneous	Multiple TCAS equipped aircraft enter each other's protection volume during flight. **Example:** random traffic threat during climb or descending flight.	All TCAS are of the same type and a known number of aircraft participate in the collision scenario. **Example:** multiple autonomous drones fly in formation.
Heterogeneous	Multiple aircraft enter each other's protection volume, but some aircraft are not TCAS equipped. **Example:** private plane with TCAS encounters an ultra-light plane.	Multiple aircraft equipped with TCAS are interacting with an aircraft without TCAS. **Example:** military aircraft intercept and escort hostile intruder.

IV. AirborneCPS: Technical Implementation

In order to simulate the functional dependency types outlined in Section I and Table I, we have implemented the conceptual functionality outlined in [10] using Laminar Research X-Plane 10[1]. The following Section IV.A provides a functional overview over TCAS. Section IV.B describes the technical implementation. Section **Error! Reference source not found.** describes the current state and feature set.

[1] http://www.x-plane.com

A. Functional Overview

Fig. 1 below shows the conceptual functions of a TCAS based on [10], which we have previously reported in [3]. A TCAS system has essentially two operation modes: cooperative and uncooperative. In cooperative mode, TCAS requests the own aircraft's transponder to sweep ("Mode-S Interrogation" in Fig. 1) the surrounding airspace for nearby aircraft ("traffic"). Cooperative traffic will reply with longitude, latitude, altitude, course, heading, and speed information ("Mode-S squitter"). In uncooperative mode, TCAS will request the own aircraft's transponder to perform a "Mode-C-Only All Call." Mode-C will reply with altitude information for any nearby traffic only. Using transponder replied, TCAS discriminates traffic and surveils possible threat targets. Both cooperative and non-cooperative traffic is either categorized as "Other Traffic," i.e. aircraft that pose no threat of collision and "Proximal Intruders," i.e. aircraft that are close and could, depending on course, altitude, and possible changes therein, become cause a collision, and "Threats," i.e. aircraft that will cause a collision, unless the pilots take immediate action. Threat classification and continuous target surveillance occurs by comparing relative movement of the traffic to the own flight path, and computing the time until traffic is intercepted ("Tau" in Fig. 1).

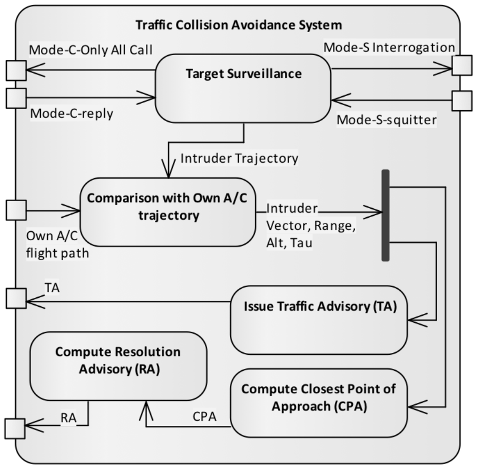
Traffic Collision Avoidance System
Mode-C-Only All Call
Mode-S Interrogation
Target Surveillance
Mode-C-reply
Mode-S-squitter
Intruder Trajectory
Comparison with Own A/C trajectory
Intruder Vector, Range, Alt, Tau
Own A/C flight path
TA
Issue Traffic Advisory (TA)
Compute Resolution Advisory (RA)
Compute Closest Point of Approach (CPA)
RA
CPA

Fig. 1. Conceptual Functions of a Traffic Collision Avoidance System

If a traffic aircraft is determined to be an intruder, a TA is issued and displayed, e.g., on the vertical speed gauge on the pilot's instrument panel (see Fig. 2). TAs as displayed as empty or filled blue diamonds ("traffic" and "proximal traffic," respectively), yellow circles ("proximal intruder") and red squares ("threat"). Positive numbers above or negative numbers below the symbol indicate vertical distance and position of the symbol represent relative to the own aircraft and its proximal space. Up- and downwards pointing arrows indicate that traffic is climbing or descending. For example, in Fig. 2, the own aircraft is climbing at 2,000ft/min. The yellow circle indicates that a proximal intruder is ca. 12nm, bearing 045, 100 feet below the own aircraft, and descending. The red square indicates that a collision is imminent from an aircraft approaching from the rear, bearing 200, 100 feet below.

If traffic is considered a threat, the closest point of approximation ("CPA" in Fig. 1) is computed and a RA is issued. Resolution advisories are overlaid on the vertical speed gage as red and green borders around needle scale (see Fig. 2). A red border indicates that, if a vertical speed of that magnitude is maintained, a collision will occur. A green border indicates the target vertical speed to maintain in order to clear the collision conflict. In Fig. 2, this means that a collision with the aircraft approaching from behind will be avoided, if a vertical speed of more than +1,500ft/min is maintained.

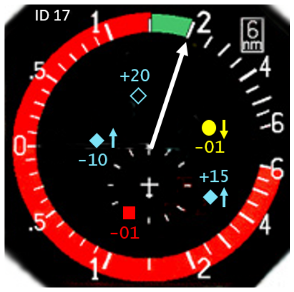

Fig. 2. Vertical Speed Gage Overlaid with TAs and an RA.

B. Technical Architecture

Fig. 3 below shows the technical architecture of AirborneCPS. As can be seen, AirborneCPS is a plugin that is executed by the *X-Plane* environment. A local area *Ethernet* connection is used to simulate Transponder information exchange. Mode-S interrogations and Mode-C all calls are submitted by the *Transponder* component of the plugin and collects information from other aircraft. This interaction implements the directed association between two or more AirborneCPS Plugins. Through the X-Plane Data Bus, information about the own aircraft's flight path (i.e. location, altitude, speed, etc.) is collected.

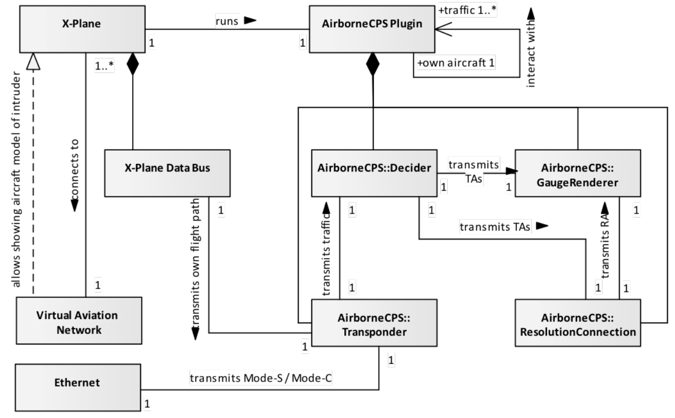

Fig. 3. Technical Architecture of the AirborneCPS plugin for X-Plane

Together with the Mode-S squitter and Mode-C reply, this information is advanced to the *Decider* component, which classifies traffic identified via the Transponder as outlined above. Results of this classifications are TAs, which are transmitted to the *GaugeRenderer* component. This component renders the gauge shown in Fig. 2. Depending on the proximity of the TA, the *ResolutionConnection* component calculates the CPA and RA based on the algorithm in [27], and instructs the

GaugeRenderer to draw the RA on the gauge. Each X-Plane instance requires its own physical hardware. Hence, each physical computer simulates one TCAS equipped aircraft. Since only Transponder messages are sent via Ethernet, a visual representation of the intruding aircraft is not accounted for. Therefore, in order to make collision avoidance scenario visually appealing, a third-party virtual aviation network[2].

V. Current State and Future directions

AirborneCPS is work in progress. Current work is concerned with implementing non-cooperative interaction scenarios as well as hostile interaction scenarios, where some intruding traffic is actively provoking a collision. At present, cooperative and non-cooperative interaction scenarios involving two or more aircraft can be simulated. Examples of its operation are available online[3]. Currently, functionality is limited to visual attenuation of TAs and Ras only. Acoustic attenuation is subject of future work. Moreover, future work will be concerned with implementing autonomous flight behavior to clear collision conflicts both through vertical and lateral separation, akin to unmanned autonomous vehicles. AirborneCPS is available[4] to interested parties.

Acknowledgments

The authors thank Laminar Research for making available their X-Plane simulator platform, their industry partners for their consultation regarding TCAS, and Travis Hafner for his consultation during implementation. The project is sponsored in part by the Federal Work Study Program of the United States.

References

[1] E. A. Lee, "Cyber Physical Systems: Design Challenges," Proc. 11th Intl. Symp. Obj. & Comp-Orien. RT Distr. Comp., 2008.

[2] W. Wolf, "Cyber-physical Systems," IEEE Comp., 42(3), 2009.

[3] W. Shanaa, S. Spier, and B. Tenbergen, "A Case Study into the Development Process of Cyber Physical Systems," Proc. 3rd Intl. WS on RE for Self-Adaptive and Cyber-Physical Systems, 2017.

[4] B. Tenbergen, M. Daun, P. Aluko Obe, and J. Brings, "View-Centric Context Modeling to Foster the Engineering of Cyber-Physical System Network," Proc. Intl. Conf. ICSA 2018.

[5] M. Daun, A. Salmon, B. Tenbergen, and T. Weyer, "Today's Challenges and Potential Solutions for the Engineering of Collaborative Embedded Systems," Proc. 2nd EITEC WS, 2015.

[6] M. Broy, M. V. Cengarle, and E. Geisberger, "Cyber-physical Systems: Imminent Challenges," Proc. 17th Monterey Conf Large-Scale Complex IT Systems, 2012.

[7] P. Mosterman and J. Zander, "Cyber-physical systems challenges: a needs analysis for collaborating embedded software systems," Softw Sys Mod 15(1), 2016, pp.5-16.

[8] J. Wan, D. Zhang, S. Zhao, L. T. Yang, and J. Lloret, "Context-aware vehicular cyber-physical systems with cloud support," IEEE Commun, 52(8), 2014 pp. 106–113.

[9] G. Schirner, D. Erdogmus, K. Chowdhury, T. Padir, "The Future of Human-in-the-Loop Cyber-Physical Systems," IEEE Comp. 46(1), 2013, pp. 36-45.

[10] US Federal Aviation Administration, "Introduction to TCAS II, Version 7.1," available at https://bit.ly/2J7Ke1C [acc. 31/05/18].

[11] J. Lee, B. Bagheri, and H.-A. Kao, "A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems," Manuf. Lett.,Vol. 3, 2015, pp. 18–23.

[12] M. Daun, J. Brings, T. Weyer, and B. Tenbergen, "Fostering concurrent engineering of cyber-physical systems," Proc. 3rd Intl. EITEC WS, 2016.

[13] P. Oliveira Antonino, A. Morgenstern, B. Kallweit, M. Becker, and T. Kuhn, "Straightforward Specification of Adaptation-Architecture-Significant Requirements of IoT-enabled Cyber-Physical Systems," Proc. 2nd Intl. WS IoT-ASAP, 2018.

[14] M. B. Gonçalves, E. Cavalcante, T. Batista, F. Oquendo, and E. Y. Nakagawa, "Towards a conceptual model for Software-intensive System-of-Systems," Proc. IEEE Intl Conf Systems, Man, and Cybernetics, 2014, pp. 1605–1610.

[15] M. Broy, "Engineering Cyber-Physical Systems: Challenges and Foundations," in Complex Systems Design & Management, Springer, Berlin, Heidelberg, 2013, pp. 1–13.

[16] P. Palensky, E. Widl, and A. Elsheikh, "Simulating Cyber-Physical Energy Systems: Challenges, Tools and Methods," IEEE Trans. Syst. Man Cybern. Syst., 44(3), 2014, pp. 318–326.

[17] S. Tripakis and R. Sengupta, "Automated Intersections: A CPS Grand Challenge," Proc. 2014 Ntl. WS Trans. CPS, available at https://cps-vo.org/node/11285 [acc 31/05/18].

[18] M. Nakamura and L. D. Bousquet, "Constructing Execution and Life-Cycle Models for Smart City Services with Self-Aware IoT," Proc. IEEE Intl Conf Autom. Computing, 2015, pp. 289–294.

[19] H. Koziolek, A. Burger, and J. Doppelhamer, "Self-commissioning Industrial IoT-Systems in Process Automation," Proc. IEEE Intl Conf Software Architecture, 2018.

[20] F. Giaimo, H. Yin, C. Berger, and I. Crnkovic, "Continuous Experimentation on Cyber-Physical Systems: Challenges and Opportunities," Proc. of the Scientific Workshop at XP2016.

[21] P. Leitão, A. W. Colombo, and S. Karnouskos, "Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges," Comput. Ind., vol. 81, 2016, pp. 11–25.

[22] H. Lin, S. Sambamoorthy, S. Shukla, J. Thorp, and L. Mili, "Power system and communication network co-simulation for smart grid applications," Proc. ISGT, 2011.

[23] X. Sun, Y. Chen, J. Liu, and S. Huang, "A co-simulation platform for smart grid considering interaction between information and power systems," Proc. ISGT, 2014.

[24] A. Al-Hammouri, "A comprehensive co-simulation platform for cyber-physical systems," Comp. Commun., 36(1), 2012.

[25] A. Al-Hammouri, V. Liberatore, H. Al-Omari, Z. Al-Qudah, M. S. Branicky, and D. Agrawal, "A Co-simulation Platform for Actuator Networks," Proc. 5th Intl. Conf Embedded Networked Sensor Systems, 2007, pp. 383–384.

[26] B. Wang and J. S. Baras, "HybridSim: A Modeling and Co-simulation Toolchain for Cyber-physical Systems," Proc. 17th EEE/ACM Intl. Symp Distr. Simulation & Real Time Appl., 2013, pp. 33–40.

[27] C. Munoz, A. Narkawicz, and J. Chamberlain, "A TCAS-II Resolution Advisory Detection Algorithm," 2013.

[2] Such as IVAO (https://ivao.aero/), VATSIM (https://www.vatsim.net/), or PilotEdge (https://www.pilotedge.net/)

[3] on Youtube: https://bit.ly/2J9EaZJ

[4] on Github: https://bit.ly/2rZfMA6