Transitioning Intelligence to Embedded Platforms

J. Mikael Eklund[1], Jonathan Sprinkle[2], Todd Templeton[2], and Shankar Sastry[2]

[1]Faculty of Engineering and Applied Science
University of Ontario Institute of Technology
2000 Simcoe Street North
Oshawa, Ontario L1H 7K4
Canada

mikael.eklund@uoit.ca

[2]Department of Electrical Engineering
and Computer Sciences,
University of California, Berkeley
Berkeley, CA 94720
USA

{sprinkle,ttemplet,sastry}@eecs.berkeley.edu

ABSTRACT

This paper addresses ongoing work with the OCP along with hardware and software to enable its use on testbeds and platforms which are better suited for embedded processors—and more affordable for research teams.

1.0 INTRODUCTION

The Open Control Platform (OCP) was instrumental in the success of the capstone demonstration of the DARPA Software Enabled Control (SEC) program [1], [2], [3], [4] and was matured by Boeing Phantom Works. The OCP is a software infrastructure which serves as a substrate for integrating innovative control technologies, and it provides comprehensive support for integrating distributed, heterogeneous components while hiding details of platform and communication from the controls developer.

The OCP enabled eight technology developer groups from academia and industry to implement advances and novel control techniques on a T-33 jet instrumented as a UCAV (Unmanned Combat Air Vehicle) surrogate with a development time of around 13 months. In the capstone demonstration formation flight [5], autonomous landing manoeuvres [6], pursuit-evasion games [7] and other experiments were successfully performed.

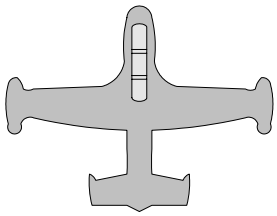
Here, large testbeds were available for the controls experiments—meaning that it was possible to take a development platform onto the aircraft. However, few (if any) academic research labs have access to testbeds which can afford to take a payload of this size and weight. This paper describes how we are enabling the OCP to be a low-overhead research tool by separating the development platform from the deployment platform, and providing example testbed implementations including a desktop simulation environment which supports both software-only and hardware-in-loop simulation.

1.1 A note on terms

In this paper the terms *platform* and *testbed* are not used interchangeably. We use the term *platform* to describe the computer architecture, operating system, and/or formfactor used to operate the controls software. Examples of platforms are: Linux, QNX, WindowsXP (operating system platforms); i386, ARM, PowerPC (hardware architecture platforms); laptop, PC-104 stack, embedded processor (form factor platforms).

Eklund, J.M.; Sprinkle, J.; Templeton, T.; Sastry, S. (2007) Transitioning Intelligence to Embedded Platforms. In *Platform Innovations and System Integration for Unmanned Air, Land and Sea Vehicles (AVT-SCI Joint Symposium)* (pp. 23-1 – 23-6). Meeting Proceedings RTO-MP-AVT-146, Paper 23. Neuilly-sur-Seine, France: RTO. Available from: http://www.rto.nato.int/abstracts.asp.

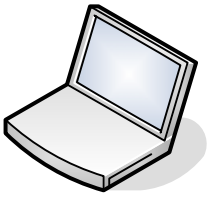
We use the term *testbed* to describe the kind of controls system upon which the software is performing. Testbeds already used by OCP include F-16, T-33, and Maverick (Helicopter). Future testbeds are described in the paper, along with how the OCP can be adapted to facilitate the use of new platforms, as well as new testbeds, by external developers.



(a) T-33 Jet Testbed

(b) Boeing Maverick (A160 Hummingbird Surrogate) Helicopter Testbed

(c) SmartBAT Testbed



(d) i386 Laptop Platform

(e) i386 PC104 Platform

(f) PowerPC Embedded Platform

Figure 1: Three testbeds for use with the OCP. Traditional OCP software execution platforms such as a laptop are appropriate for testbeds such as 1(a) and 1(b), but are too heavy for testbeds such as 1(c). Note that testbeds and platforms are scaled relatively, not globally

2.0 THE OPEN CONTROL PLATFORM

The OCP is a software infrastructure designed to assist in the development and testing of control algorithms targeted for execution in embedded software.

The successful integration, testing, and demonstration of the SEC technology for the Capstone Demonstration was enabled by the use of OCP, since it provided a layer of abstraction from the vehicle control and avionics. This allowed the technology developers to integrate their control algorithms onto testbeds whose avionics were classified or ITAR restricted without having access to those testbeds, and it allowed the Boeing developers to create the testbed interfaces (avionics and system) within the OCP architecture.

Furthermore, the OCP could be interfaced with a desktop simulation program to allow for thorough software simulation by the application developers prior to the hardware-inthe- loop testing performed by the system integrators. This made it possible to develop and perform the necessary testing prior to the flight tests on the T-33 testbed.

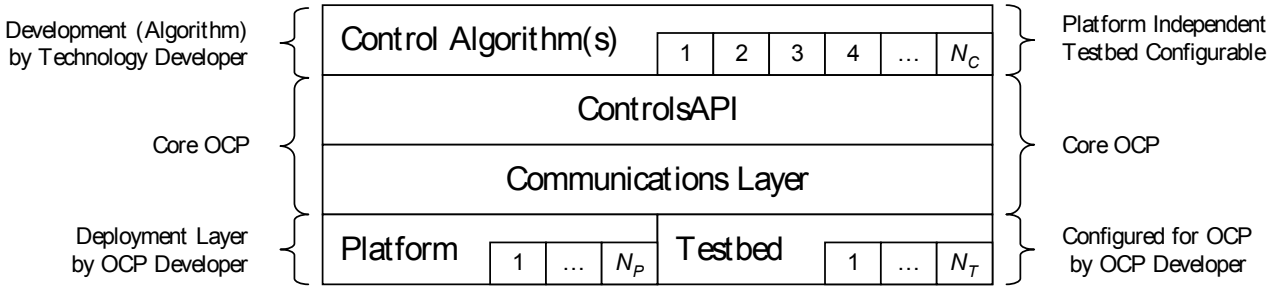


Figure 2: Development and deployment layers of the OCP. Note that the control algorithms may now be developed on a separate platform from the deployment platform. Each of the N_C control algorithms are configured to run on one or more of the N_T testbeds, and can be written to run on *any* of the N_P platforms.

As a core technology, the OCP provides an interface to describe software tasks which are either periodic or event-driven. These software tasks implement control strategies such as Model-Predictive Control [7], performing reach-avoid calculations [5], or executing formation flight [8]. In order to take advantage of this core, it is necessary to consider how to develop and deploy those technologies.

The transparency which enabled avionics to be hidden also allows for independent development of new testbeds and platforms. Those testbeds and platforms must be integrated (nontrivially) by an OCP developer. To enable those who are *not* software experts to perform this integration is facilitated by examples to be used as templates.

2.1 Development Improvements

A design goal of OCP is to reduce the complexity of using existing control algorithms on different testbeds, and executing on different platforms. Technology developers (i.e., those who are writing controls algorithms through the ControlsAPI) will be able to download the OCP as a development environment. These users will be able to implement controls strategies on existing testbeds. OCP developers (i.e., those who are writing testbed interfaces or platform interfaces) will be able to reference existing testbed and platform configurations for development of new ones. Open-source avionics for off-the-shelf testbeds will make configuration of new testbeds more transparent.

2.2 Deployment Improvements

One major problem with deploying autonomous systems today is that the development platform and deployment platform are often the same. For small-footprint devices (meaning that there is a small amount of disk space and/or memory available) it is impossible to build the the control software *on that device*. Our improvements to the deployment process will allow users on any supported development platform to build the deployed OCP executable *on the development machine*. This will allow very small embedded platforms (e.g., PXA255) with storage space on the order of tens of megabytes to execute OCP control algorithms (an order of magnitude reduction in space requirements).

For our testbed additions, we are using the UAVs in the Berkeley Aerial Robotics (BEAR) lab to develop avionics interfaces for a variety of testbeds, including Yamaha R-50 and RMax helicopters, electric helicopters and the fixed-wing Berkeley SmartBAT which is based on the ZAGI flying-wing.

The SmartBAT UAV in particular has a payload of about 900 grams, but is inexpensive enough to provide a very practical platform for UAV research.

As part of this project and in the first testbed application, the SmartBAT UAV, we are using the commercially available CrossBow MNAV system to provide the hardware interface to the SmartBAT, basic flight controller and safety pilot authority of the testbed [9], [10]. The MNAV system is supported by an open source autopilot and ground station system [11]. We have additionally integrated a desktop simulator into the system, which is based on the open source CRRCSim simulator, which was developed for model airplanes including the ZAGI flying-wing testbed [12]. A screenshot of this simulator in operation is shown in Figure 3.

Figure 3: A screenshot of the modified CRRCSim simulator running on the same desktop computer as the MNAV autopilot ground station system. The ZAGI aircraft can be seen in the distance above the clouds in the upper right window, and as the red and yellow swept-wing shape near the upper right waypoint in the ground station viewer.

In each case the necessary interfaces are being incorporated into the open source releases of the projects. At this time they have been fully included in the CRRCSim project, however only in a precompiled Windows binary release for the MNAV Autopilot.

This desktop simulation can be operated as a software system or a hardware-in-the loop simulation with the MNAV hardware system, allowing a similar development environment as provided in the SEC program for OCP application developers. To allow the software-only simulation, an interface was developed for CRRCSim which emulates the Crossbow Technology MNAV vehicle sensor suite and servo control board.

This simulation system will be incorporated into the OCP system as the simulation environment for this testbed, and it will be applied to other testbeds that can be operated with the CRRCSim system and MNAV hardware.

The OCP has proven itself to be a very useful and effective tool for control development on UAV systems. However its use and availability has been largely limited to SEC program participants due to limited testbed availability and complexity of adding to the core interfaces. Our team is working to provide an Open Source and easily accessible and usable version of OCP such that it is available to all in the controls research and development community.

Prior to the conference dates, the OCP development, deployment, core, and technology developer sources and binaries will be available through the ESCHER Research Institute. Portions of the system are currently available through the CRRCSim and MNAV Autopilot releases. This release will include increased configurability and demonstration examples for its use with common processing platforms and UAV testbeds. In our presentation will be providing detailed information on how to obtain these releases.

3.0 CONCLUSION

The contribution of this work is to reduce the overhead required for an institution with interesting algorithms for autonomous behavior to step into the domain of testing. At this time, investment in the hundreds of thousands of dollars (for rotorcraft) is required in order to carry a modest computing platform.

Our work has been to separate the archaic build process from the execution platform, opening up the computational venue to machines which are closer to embedded platforms than general computing platforms, and more accessible to researchers, and to integrate a desktop simulation environment for small UAVs.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the generous contributions of Dr. David H.C. Shim, Boeing Phantom Works and the ESCHER Institute, particularly the technical assistance provided by the Boeing team of Brian Mendel, Jared Rosson, Dr. Doug Stuart, James L. Paunicka and Dr. David E. Corman.

RELEASE CONDITIONS

The information contained in the abstract, and to be contained in the full paper, carries unlimited and unrestricted distribution rights, given that the authors are appropriately credited. In addition, there are no rules or restrictions governing presentation of this material during the workshop.

REFERENCES

[1] B. Heck, L. Walls, and G. Vachtsevanos, "Software Enabled Control: Background and motivation," in *Proceedings of the 2001 American Control Conference*, vol. 5. IEEE/AACC, June 2001, pp. 3433–3438.

[2] J. L. Paunicka, B. R. Mendel, and D. E. Corman, "The OCP–an open middleware solution for embedded systems," in *Proceedings of the 2001 American Control Conference*, vol. 5. IEEE/AACC, June 2001, pp. 3445–3450.

[3] J. L. Paunicka, D. E. Corman, and B. R. Mendel, "A CORBA-based middleware solution for UAVs," in *Proceedings of the Fourth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC–2001)*, May 2001, pp. 261–267.

[4] L. Wills and et al., "An open platform for reconfigurable control," *IEEE Control Systems Magazine*, vol. 21, pp. 49–64, June 2001.

[5] D. M. Stipanovic, G. Inalhan, R. Teo, and C. J. Tomlin, "Decentralized overlapping control of a formation of unmanned aerial vehicles," in *Processing of the 41st IEEE Conference on Decision and Control (CDC)*, vol. 3. IEEE, Dec. 2002, pp. 2829–2835.

[6] J. Sprinkle, A. D. Ames, J. M. Eklund, I. Mitchell, and S. S. Sastry, "Online safety calculations for glideslope recapture," *Innovations in Systems and Software Engineering*, vol. 1, no. 2, pp. 157–175, Sept. 2005.

[7] J. M. Eklund, J. Sprinkle, and S. S. Sastry, "Implementing and testing a nonlinear model predictive tracking controller for aerial pursuit evasion games on a fixed wing aircraft," in *Proceedings of American Control Conference (ACC) 2005*, June 2005, pp. 1509–1514.

[8] F. Borrelli, T. Keviczky, and G. J. Balas, "Collision-free UAV formation flight using decentralized optimization and invariant sets," in *Proceedings of the 43rd IEEE Conference on Decision and Control*, vol. 1. IEEE, Dec. 2001, pp. 1099–1104.

[9] J. S. Jang and D. Liccardo, "Automation of small uavs using a low cost mems sensor and embedded computing platform," in *25th Digital Avionics Systems Conference*, Oct. 2006.

[10] Crossbow Technology Inc., "MNAV100CA calibrated digital sensor and servo control system," n.d. [Online]. Available: http://sourceforge.net/projects/micronav

[11] J. S. Jang (project administrator), "MNAV autopilot," n.d. [Online]. Available: http://sourceforge.net/projects/micronav

[12] J. E. Kansky (project manager) et al., "CRRCsim: A model airplane flight simulation program," n.d. [Online]. Available: http://crrcsim.sourceforge.net/