Project Proposal

Daniel McInnes

1 Abbreviations

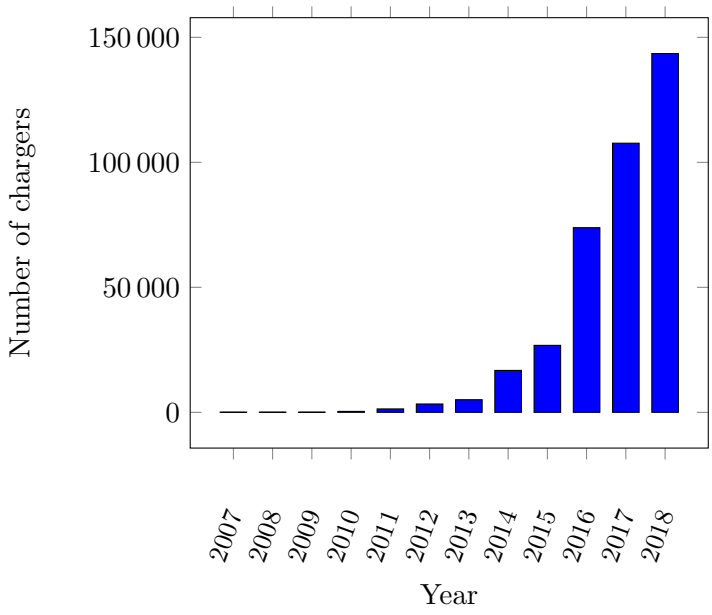
CSMS: Charging Station Management System: manages Charging Stations and has the information for authorizing Users for using its Charging Stations.
JVM: Java Virtual Machine
OCA: Open Charge Alliance
OCPP: Open Charge Point Protocol

2 Topic Definition

2.1 Background

Currently, networks of publicly available electric vehicle fast chargers communicate with servers using the Open Charge Point Protocol (OCPP). Ideally, the software running on these servers would be error free and never crash. Numerous software verification tools exist to prove desirable properties of the server software, such as functional correctness, the absence of race conditions, memory leaks, and certain runtime errors.

The International Energy Agency [1] reports that the number of publicly available fast chargers (> 22kW) increased from 107 650 in 2017 to 143 502 in 2018.



The Open Charge Alliance [69] reports that more than 10 000 charging stations[1] in over 50 countries are managed using the OCPP protocol.

2.2 Project Goal

The goal of this project is to develop an OCPP v2.0 server for use in electric vehicle charger networks. The server software should verifiably have the following desirable properties:

- The program should not leak memory.
- The program should never access uninitialized memory (for example, should never read past the end of an array).
- The program should never crash or exit unexpectedly.
- The program should be bounded in terms of memory (RAM) usage at runtime.
- The program should be bounded in terms of the time taken to complete operations, i.e. should never 'hang'.

[1] Note that a "charging station" may consist of multiple fast chargers, thus the disparity between 143 502 "fast chargers" and "more than 10 000 charging stations"

- The program should verifiably meet its requirements. These requirements are typically in the form of preconditions and postconditions.
- The program should never exhibit undefined behaviour.
- The program should constrain information flow, i.e. not leak sensitive information such as passwords.

2.3 Criteria for the Selection of Software Verification Tools

Several automated software verification tools are currently available. The features of these tools will be evaluated and a suitable candidate selected for the implementation of the server. The criteria for the selection is as follows:

- The tool should verify the properties listed in the previous section.
- The tool should be sound. Many of the tools claim to be sound "modulo bugs in the tool", and have lengthy lists of known bugs.
- There are many cases where a verification tool cannot prove or disprove properties of a piece of software within a certain timeframe. Preference will be given to verification tools that perform better in this respect.
- Preference will be given to verification tools with Evidence of successful use in commercial software development.
- Preference will be given to languages with existing libraries.
- Preference will be given to verification tools with permissive licenses.
- Preference will be given to verification tools that support multiple platforms (eg. Windows, Linux).
- Preference will be given to verification tools that detect data races in multithreaded applications.

2.4 A Minimal OCPP v2.0 Implementation

The OCPP v2.0 specification covers many use cases that are not essential for the implementation of a basic CSMS. The goal of this project is to implement a basic subset, as defined in [70, p.10].

This includes booting a charge station (use cases B01-B04), configuring a charge station (B05-07), resetting a charge station (B11-B12), authorization

options (C01), a transaction mechanism (E01), availability (G01), monitoring events (G05, N07), sending transaction related meter values (J02), and data transfer(P01, P02)

3 Review of Background and Related Work

Numerous software verification tools were considered for use in implementing the OCPP server. These tools include AdaSPARK, Dafny, KeY, OpenJML, Spec#, VCC, Verifast, Viper, Whiley, and Why3.

3.1 SPARK 2014

SPARK 2014 is both a formally defined programming language and a set of verification tools. In typical use, a programmer writes SPARK code, which is compiled by the GNAT compiler, then analyzed by the GNATprove tool to produce numerous verification conditions.

GNATprove uses Alt-Ergo (OCamlPro, 2014), CVC4 (NYU, 2014), YICES (Dutertre, 2014) and Z3 (Bjorner, 2012) to prove the verification conditions.

3.1.1 Features

Formally verifies:

- information flow
- freedom from runtime errors
- functional correctness

3.1.2 Safety Standards

SPARK 2014 satisifes:

- DO-178B/C
- Formal Methods supplement DO-333
- CENELEC 51028
- IEC 61508
- DEFSTAN 00-56

3.1.3 Soundness

SPARK is thought to be sound. Internet searches failed to find examples of unsoundness.

3.1.4 Supported Platforms

Windows, Linux, OSX.

3.1.5 License Information

Dual license:

SPARK GPL is available for free from http://libre.adacore.com under the GPL.

SPARK PRO is available under a commercial license from http://www.adacore.com.

3.1.6 Evidence of successful use in commercial software development

There is abundant evidence of the successful use of SPARK in high integrity software development. See: [31], [28], [29], [2], [8], [23], [39], [5], [38], [17], [4], [10], [44], [25], [32], [37], [34], [22], [21], [19], [35], [36], [43], [24], [12], [33], [20], [14], [40], [26], [27], [42], [9], [46], [30], [7], [11], [47], [13], [6], [16], [15], [45], [18], [3].

Of particular relevance is the experience of the CubeSat Laboratory at Vermont Technical College[41]. Cubesats are small cubes launched into space with various sensors onboard, in this case without post-launch software update capabilities. This means the software must be fault free at the time of launch. The students (mostly third and fourth year undergraduates, with no prior knowledge of SPARK or Ada, and a high turnover rate) proved the software to be free of runtime errors. 14 Cubesats were launched in November 2013. Most were never heard from again, but the SPARK Cubesat worked for 2 years until it reentered Earth's atmosphere as planned in November 2015.

3.1.7 Existing Libraries

SPARK has a minimal container library. SPARK interfaces easily with Ada, which has an extensive standard library.

3.1.8 Multithreaded Application Support

Unsupported.

3.1.9 Supported Languages

SPARK 2104 supports a subset of Ada 2012.

See [63, p.18]

"The following Ada 2012 features are not currently supported by Spark:

Aliasing of names; no object may be referenced by multiple names

Pointers (access types) and dynamic memory allocation

Goto statements

Expressions or functions with side effects

Exception handlers

Controlled types; types that provide fine control of object creation, assignment, and destruction

Tasking/multithreading (will be included in future releases)"

3.2 Dafny

3.2.1 Home Page

https://rise4fun.com/Dafny

3.2.2 Features

Dafny is both a language and a verifier [62]. Dafny supports feature verification via preconditions, postconditions, loop invariants and loop variants. It uses the 'Boogie' intermediate language and the Z3 theorem prover. The Dafny compiler produces executable files for the .NET platform [64].

3.2.3 Soundness

Dafny is designed to be sound but incomplete, and is known to report errors on correct programs [65].

3.2.4 Supported Platforms

Windows, Linux, OSX host, for a .NET target platform.

3.2.5 License Information

MIT

3.2.6 Evidence of successful use in commercial software development

Internet searches failed to find any evidence of Dafny being used in commercial software development.

3.2.7 Existing Libraries

There is a 'mathematics' library for Dafny.

3.2.8 Multithreaded Application Support

Internet searches failed to find evidence of multithreaded application support.

3.2.9 Supported Languages

Dafny.

3.3 KeY

3.3.1 Home Page

https://www.key-project.org/

3.3.2 Features

KeY offers functional verification for Java programs. The specifications are written as comments in JML in the Java source code. KeY is built on a formal logic called 'Java Card DL', which is itself a first-order dynamic logic, and an extension of Hoare logic. It is targeted at JavaCard programs.

3.3.3 Soundness

KeY is thought to be sound. Internet searches failed to find examples of unsoundness.

3.3.4 Supported Platforms

Windows, Linux, and OSX hosts, for a JVM target.

3.3.5 License Information

GPL.

3.3.6 Evidence of successful use in commercial software development

Internet searches failed to find any evidence of KeY being used in commercial software development.

3.3.7 Existing Libraries

There are extensive libraries available for Java programs.

3.3.8 Multithreaded Application Support

No.

3.3.9 Supported Languages

Java.

3.4 OpenJML

3.4.1 Home Page

http://www.openjml.org/

3.4.2 Features

OpenJML is a suite of tools for verifying Java programs that are annotated with JML statements. It is based on OpenJDKv1.8. It detects illegal memory access at compile time. It verifies preconditions and postconditions. It arguably guarantees the absence of undefined behaviour for single threaded applications. It does not constrain information flow.

3.4.3 Soundness

Yes

3.4.4 Supported Platforms

Windows, Linux, OSX.

3.4.5 License Information

GPLv2.

3.4.6 Evidence of successful use in commercial software development

Internet searches failed to find any evidence of OpenJML being used in commercial software development.

3.4.7 Existing Libraries

There are extensive libraries available for Java programs.

3.4.8 Multithreaded Application Support

No.

3.4.9 Supported Languages

Java (only OpenJDK v1.8, may become unsupported in December 2020)

3.5 Spec#

https://www.microsoft.com/en-us/research/project/spec/

3.5.1 Features

Spec# consists of the Spec# programming language, the Spec# compiler, and the Spec# static program verifier. The programming language is an extension of C#, adding non-null types, checked exceptions, preconditions, postconditions, and object invariants [66]. It uses Boogie for verification.

3.5.2 Soundness

Spec# claims to be sound[50].

3.5.3 Supported Platforms

Windows host platform, .NET target platform.

3.5.4 License Information

Internet searches failed to find Spec# license information.

3.5.5 Evidence of successful use in commercial software development

Internet searches failed to find evidence of Spec# being used in commercial software development.

3.5.6 Existing Libraries

Internet searches failed to find Spec# libraries. However, Spec# offers interoperability with the .NET platform, which has an extensive standard library, although the soundness of this library is not guaranteed.

3.5.7 Multithreaded Application Support

Yes[66].

3.5.8 Supported Languages

Spec#.

3.6 VCC

3.6.1 Home Page

https://www.microsoft.com/en-us/research/project/vcc-a-verifier-for-concurrent-c/

3.6.2 Features

VCC verifies preconditions and postconditions written in the form of special comments in the source code. It detects data races in multithreaded applications and illegal memory access at compile time. It does not guarantee the absence of undefined behaviour.

3.6.3 Soundness

Yes

3.6.4 Supported Platforms

Windows

3.6.5 License Information

MIT [68]

3.6.6 Evidence of successful use in commercial software development

VCC has been used successfully in at least one major commercial project, the verification of the Microsoft Hypervisor, the virtualization kernel of Hyper-V [53].

3.6.7 Existing Libraries

Internet searches failed to find libraries verified with VCC. However, applications written with VCC can link against unverified libraries, effectively giving access to extensive library support.

3.6.8 Multithreaded Application Support

Yes

3.6.9 Supported Languages

C

3.7 Verifast

3.7.1 Home Page

https://github.com/verifast/verifast

3.7.2 Features

Verifast verifies preconditions, and postconditions in the form of special comments in the source code. It detects race conditions in multithreaded applications. It does not guarantee the absence of undefined behaviour, or verify the absence of stack overflows.

3.7.3 Soundness

No, see https://github.com/verifast/verifast/blob/master/soundness.md

3.7.4 Supported Platforms

Windows, Linux, OSX.

3.7.5 License Information

MIT [61].

3.7.6 Evidence of successful use in commercial software development

There is evidence of some use in industrial applications[77].

3.7.7 Existing Libraries

Internet searches failed to find libraries verified by / written with VeriFast annotations. However, applications written with VeriFast can link against unverified libraries, effectively gaining access to extensive library support.

3.7.8 Multithreaded Application Support

Yes.

3.7.9 Supported Languages

C, Java

3.8 Viper

3.8.1 Home Page

https://www.pm.inf.ethz.ch/research/viper.html

3.8.2 Features

Viper consists of the Viper intermediate verification language, automatic verifiers, and example front end tools [57]. It is more of a tool for creating other verification tools than a tool for verifying software. In practice, a developer would write code in Python and use the 'Nagini' front end (based on Viper) to verify the code [55]. A corresponding front end for Rust exists, 'Prusti' [49].

3.8.3 Soundness

Yes.

3.8.4 Supported Platforms

Windows, MacOS, Linux[60].

3.8.5 License Information

https://bitbucket.org/viperproject/carbon/src/default/LICENSE.txt Mozilla Public License Version 2.0[56]

3.8.6 Evidence of successful use in commercial software development

Internet searches failed to find evidence of Viper being used in commercial software development.

3.8.7 Existing Libraries

Internet searches failed to find libraries verified by Viper front ends. However, applications written with these front ends can link against unverified libraries, effectively gaining access to extensive library support.

3.8.8 Multithreaded Application Support

Yes.

3.8.9 Supported Languages

Python, Rust, Java, OpenCL, Chalice.

3.9 Whiley

3.9.1 Home Page

http://whiley.org/

3.9.2 Features

Whiley consists of the Whiley language, the Whiley Build System, the Whiley Compiler, the Whiley Intermediate Language, the Whiley-2-Java Compiler, the Whiley-2-C Compiler, and the Whiley Constraint Solver[76].

Whiley uses a variant of first-order logic called the Whiley Assertion Language for verification.

In typical use, a developer will write source code in Whiley, build with the Whiley Build system, and execute the resulting Java class file on the JVM.

3.9.3 Soundness

Internet searches did not find evidence to say that Whiley is unsound.

3.9.4 Supported Platforms

JVM.

3.9.5 License Information

BSD.

3.9.6 Evidence of successful use in commercial software development

Internet searches failed to find evidence of Whiley being used in commercial software development.

3.9.7 Existing Libraries

Internet searches failed to find libraries verified by Whiley. However, applications written with Whiley can link against unverified Java functions, effectively gaining access to extensive library support.

3.9.8 Multithreaded Application Support

No.

3.9.9 Supported Languages

Whiley. The Whiley-2-Java Compiler (WyJC) can convert verified Whiley programs into JVM class files. Whiley can import Java functions, and export Whiley functions for use in Java programs.

3.10 Why3

3.10.1 Home Page

http://why3.lri.fr/

3.10.2 Features

The Why3 deductive program verification platform includes the WhyML language, a standard library of logical theories, and basic programming data structures[83]. In typical use, a developer will write software in WhyML, and get correct-by-construction OCaml programs through an automated extraction mechanism. It verifies preconditions and postconditions.

WhyML is also used by numerous popular verification tools (FramaC, SPARK2014, Krakatoa) as an intermediate language for verification of C, Java and Ada programs.

3.10.3 Soundness

Yes.

3.10.4 Supported Platforms

Windows, Linux, OSX. Why3 is distributed as a Debian package and as an OPAM package.

3.10.5 License Information

GNU LGPL 2.1.

3.10.6 Evidence of successful use in commercial software development

Internet searches failed to find evidence of WhyML being used in commercial software development. However, there are countless cases of commercial software development using WhyML as an intermediate language, as it is used by FramaC, SPARK, and others.

3.10.7 Existing Libraries

Why3 comes with a standard library of logical theories and basic programming data structures.

3.10.8 Multithreaded Application Support

No.

3.10.9 Supported Languages

WhyML.

3.11 Conclusion of Review of Background and Associated Work

Based on the properties of the verification tools available, it has been decided to use SPARK 2014 for the implementation and verification of the OCPP server. It has wide use in industry, which gives me confidence that it is a practical choice. It verifies all of the properties that are important to me, and has good library support.

4 Risk Assessment

4.1 OHS

The student will work in a home office, covered by general OHS laboratory rules. Although the project is related to charging electric vehicles, there is no requirement for exposure to high voltages. The project is confined to the development of a transaction processing server.

4.2 Other Risks

4.2.1 Project risks

The goal of this project is to create a verified OCPP v2.0 server. Even if this goal is met, there is a chance that the OCPP v2.0 protocol never achieves acceptance in the field, which would render the server useless. There is no practical mitigation for this risk.

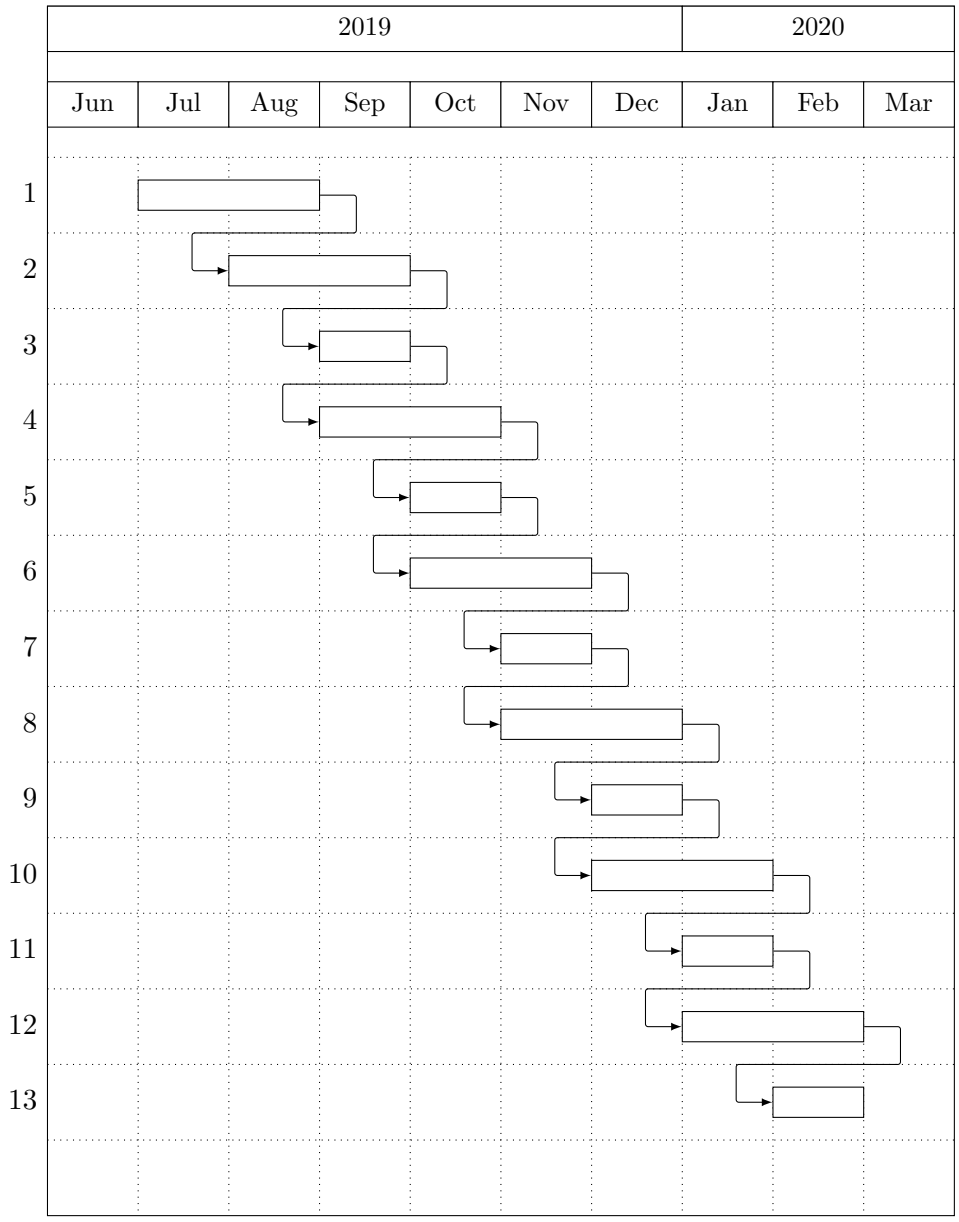
4.2.2 Technology risks

A decision has been made to use SPARK 2014. There is no real support available in the event of technical issues. The project supervisor has no experience with SPARK. The student has minimal experience with SPARK. However, there are many resources available online.

4.2.3 Scheduling risks

Due to a lack of experience in the development of verified software, the student has nothing to base task estimates on. The timeline associated with feature delivery is based on nothing more than a gut feeling.

5 Project Plan



- 1. Review the literature of the verification tools, and choose one based on the selection criteria.

  Duration: 8 days. 2019-07-02 → 2019-08-16.

- 2. Get an Ada websocket server running, and a basic websocket client connecting to it.

  Duration: 4 days. 2019-08-1 → 2019-09-01.

- 3. Modify the websocket client to request upgrading the websocket connection to an OCPP connection.

  Duration: 2 days. 2019-09-01 → 2019-09-15.

- 4. Modify the websocket server to accept websocket connection 'upgrade to OCPP' requests.

  Duration: 2 days. 2019-09-15 → 2019-10-01.

- 5. Modify the OCPP server to accept and respond to 'Boot Notification' requests.

  Duration: 2 days. 2019-10-01 → 2019-10-14.

- 6. Modify the OCPP server to accept and respond to 'Get Variable' and 'Set Variable' requests.

  Duration: 2 days. 2019-10-14 → 2019-11-01.

- 7. Modify the OCPP server to handle 'Reset' requests and responses.

  Duration: 2 days. 2019-11-01 → 2019-11-15.

- 8. Modify the OCPP server to handle 'Authorize' requests and responses.

  Duration: 2 days. 2019-11-15 → 2019-12-01.

- 9. Modify the OCPP server to handle 'TransactionEvent' requests and responses.

  Duration: 2 days. 2019-12-01 → 2019-12-15.

- 10. Modify the OCPP server to handle 'StatusNotification' requests and responses.

  Duration: 2 days. 2019-12-15 → 2020-01-01.

- 11. Modify the OCPP server to handle 'NotifyEvent' requests and responses.

  Duration: 2 days. 2020-01-01 → 2020-01-15.

- 12. Modify the OCPP server to handle 'TransactionEvent (Meter Values)' requests and responses.

  Duration: 2 days. 2020-01-15 → 2020-02-01.

- 13. Modify the OCPP server to handle 'DataTransfer' requests and responses. Duration: 2 days. 2020-02-01 → 2020-02-16.

References

[1] "Global EV Outlook 2019," International Energy Agency, Paris, France, Tech. Rep., May 2019, Accessed: June 25, 2019. [Online]. Available: https://webstore.iea.org/global-ev-outlook-2019

[2] "Ada on Board: GNAT Pro Helps ExoMars Get to the Red Planet". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/ada-on-board-gnat-pro-helps-exomars-get-to-the-red-planet

[3] "Ada on Board: Thales Using AdaCores GNAT Pro for Critical Avionics Software". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/ada-on-board-thales-using-adacores-gnat-pro-for-critical-avionics-software

[4] "AdaCore and Altran Toolsets Help Launch CubeSat into Orbit". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/cubesat

[5] "AdaCore Development Environment Selected for New Spanish Satellite Project". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/spanish-satellite-project

[6] "AdaCore Enhances Security-Critical Firmware with NVIDIA". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/adacore-enhances-security-critical-firmware-with-nvidia

[7] "AdaCore Helps AAI Upgrade the T25 SECT Electronic Combat Trainer". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/adacore-helps-aai-upgrade-the-t25-sect-electronic-combat-trainer

[8] "AdaCores CodePeer Selected for Digital Terrain System Requiring DO-178B Certification". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/codepeer-do178b-certification

[9] "Astrium in the UK Selects GNAT Pro for Environmental Satellite System". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/customers/astute-class-submarine-periscope

[10] "Astrium Selects AdaCores GNAT Pro and PolyORB for International Space Station". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/astrium-polyorb

[11] "AVIO Selects AdaCores GNAT Pro Assurance Toolsuite for European Space Agency Program". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/avio-selects-adacores-gnat-pro-assurance-toolsuite-for-european-space-agency-program

[12] "Barco selects GNAT Pro for Advanced Avionics Applications". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/barco

[13] "Case Study BAE Systems Eurofighter Typhoon". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/customers/eurofighter-typhoon

[14] "Case Study EADS CASA In-flight Refuelling Boom System". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/uploads/customers/CaseStudy_Boom.pdf

[15] "Case Study MDA - Canadian Space Arm". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/uploads/customers/CaseStudy_SpaceArm.pdf

[16] "Case Study Pilot Ejection Seat". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/customers/pilot-ejection-seat

[17] "Deep Blue Capital Selects AdaCore Products for Financial System Development". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/deep-blue-capital-financial-system-development

[18] "DENSO Using SPARK Technology for Automotive Research Project". AdaCore. Accessed: Aug 17, 2019.

[19] "Digicomp Shows Continuing Success with Ada and GNAT Pro". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/digicomp-success

[20] "EADS CASA Selects AdaCore Toolset for nEUROn Unmanned Aircraft". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/neuron-unmanned-aircraft

[21] "Embraer Selects Ada and AdaCores GNAT Pro for AMX Upgrade". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/embraer-amx-upgrade

[22] "Eurocopter Selects GNAT Pro for Military Helicopter ARINC 653 Project". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/arinc-653-project

[23] "French Agency DGA Selects AdaCores GNAT Pro with SQUORE Technology". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/dga-gnat-pro-squore-technology

[24] "GNAT Pro Chosen for UKs Next Generation ATC System". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/adacore-gnat-pro-chosen-for-uk-next-generation

[25] "GNAT Pro Safety-Critical used by Terma A/S for Space Monitor Project". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/terma

[26] "Hamilton Sundstrand Selects GNAT Pro For Boeing 787 Air Conditioning Control Unit". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/hamilton-sundstrand-boeing-air-conditioning

[27] "Lockheed Martin Selects GNAT Pro for C-130J Software". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/a350

[28] "MDA Selects AdaCores GNAT Pro Assurance Development Platform for International Space Station Software". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: www.adacore.com/press/mda-gnatpro-space-station

[29] "MHI Aerospace Systems Corp. Selects AdaCores QGen for Model-Based Development". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/mhi-aerospace-systems-qgen

[30] "Protecting Navy Vessels with Ship Anti-Air Warfare Capability". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.raytheon.com/capabilities/products/ssds

[31] "Revolutionary Artificial Heart". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/scandinavian-real-heart-selects-adacore-embedded-software-development-platform-for-revolutiona

[32] "Rockwell Collins Develops SecureOne with SPARK Pro and GNAT Pro High-Security". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/secureone

[33] "Rockwell Collins Selects GNAT Pro for Advanced Avionics Display System". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/rockwell-avionics-display

[34] "Saab Electronic Defence Systems Adopts CodePeer". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/saab-electronic-defence-systems-adopts-codepeer

[35] "Siemens Switzerland Selects AdaCore Toolset for Railway Project siemens-railway". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/siemens-railway

[36] "Singo Solution". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.contactcenterworld.com/view/contact-center-case-study/singo-solution.aspx

[37] "SmartSide Adopts Ada and GNAT Pro for Smart Devices Platform". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/smartside-adopts-ada-gnat-pro

[38] "SmartWard Pty Ltd Selects AdaCore Tools for Hospital Information System Development". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/smartward-hospital-information-system

[39] "SPARK Going to the Moon". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/spark-going-to-the-moon

[40] "SPARK Pro Adopted by secunet". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/spark-pro-secunet

[41] "ten years of using spark to build cubesat nano satellites with students". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://blog.adacore.com/ten-years-of-using-spark-to-build-cubesat-nano-satellites-with-students

[42] "Thales Aerospace Division Selects GNAT Pro for Airbus A350 XWB (Xtra Wide-Body)". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/astrium

[43] "Thales Selects AdaCore Toolset for Argos Satellite Project". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/thales-argos-satellite

[44] "TOYOTA ITC Japan Selects SPARK Pro Language and Toolset for High-Reliability Research Project". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/toyota-itc-japan-selects-spark-pro-language-and-toolset-for-high-reliabilit

[45] "University of Colorados Laboratory for Atmospheric and Space Physics adopts Ada and GNAT Pro for NASA project". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/lasp-selects-gnat-pro-for-clarreo

[46] "Wind River Powers New Generation of Periscope on Royal Navys Next-Generation Astute Class Submarines". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/uploads/customers/adacore_casestudy_uret.pdf

[47] "Wind River Teams with AdaCore on Safety-Critical ARINC 653 for use in Boeing 7E7". AdaCore. Accessed: Aug 17, 2019. [Online]. Available: https://www.adacore.com/press/arinc-653

[48] W. Ahrendt, B. Beckert, R. Bubel, R. Hähnle, P. H. Schmitt, and M. Ulbrich, Eds., *Deductive Software Verification - The KeY Book: From Theory to Practice*, ser. Lecture Notes in

Computer Science. Springer, 2016, vol. 10001. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-49812-6

[49] V. Astrauskas, P. Müller, F. Poli, and A. J. Summers, "Leveraging Rust types for modular specification and verification," ETH Zurich, Tech. Rep., 2019.

[50] M. Barnett, K. R. M. Leino, and W. Schulte, "The Spec# programming system: An overview," in *Construction and Analysis of Safe, Secure, and Interoperable Smart Devices: International Workshop, CASSIS 2004, Marseille, France, March 10-14, 2004, Revised Selected Papers*, ser. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, vol. 3362, pp. 49–69.

[51] F. Bobot, J.-C. Filliâtre, C. Marché, and A. Paskevich, "Why3: Shepherd Your Herd of Provers," in *Boogie 2011: First International Workshop on Intermediate Verification Languages*, Wroclaw, Poland, 2011, pp. 53–64. [Online]. Available: https://hal.inria.fr/hal-00790310

[52] E. Cohen, M. Hillebrand, S. Tobies, M. Moskal, and W. Schulte, "Verifying C Programs: A VCC Tutorial," University of Freiburg, Leuven, Belgium, Tech. Rep., July 10 2015, accessed: June 22, 2019. [Online]. Available: https://swt.informatik.uni-freiburg.de/teaching/SS2015/swtvl/Resources/literature/vcc-tutorial-col2.pdf

[53] E. Cohen, S. Tobies, M. Moskal, and W. Schulte, "A Practical Verification Methodology for Concurrent Programs," Microsoft Research, 1 Microsoft Way, Redmond, WA, Tech. Rep., February 12 2009, accessed: June 22, 2019. [Online]. Available: https://www.microsoft.com/en-us/research/publication/a-practical-verification-methodology-for-concurrent-programs/

[54] D. Cok, "OpenJML: Software verification for Java 7 using JML, OpenJDK, and Eclipse," vol. 149. Open Publishing Association, 2014, pp. 79–92.

[55] M. Eilers and P. Müller, "Nagini: A static verifier for python," in *CAV*, 2018.

[56] "Carbon License". ETH Zurich. Accessed: Aug 18, 2019. [Online]. Available: https://bitbucket.org/viperproject/carbon/src/default/LICENSE.txt

[57] "Viper". ETH Zurich. Accessed: Aug 18, 2019. [Online]. Available: https://www.pm.inf.ethz.ch/research/viper.html

[58] J.-C. Filliâtre and A. Paskevich, "Why3 – Where Programs Meet Provers," in *ESOP'13 22nd European Symposium on Programming*, ser. LNCS, vol. 7792. Rome, Italy: Springer, Mar. 2013. [Online]. Available: https://hal.inria.fr/hal-00789533

[59] B. Jacobs, J. Smans, and F. Piessens, "The Verifast Program Verifier: A Tutorial," Department of Computer Science, Katholieke Universiteit, Leuven, Belgium, Tech. Rep., November 2017, Accessed: June 20, 2019. [Online]. Available: https://people.cs.kuleuven.be/~{}bart.jacobs/verifast/tutorial.pdf

[60] U. Juhasz, I. T. Kassios, P. Mller, M. Novacek, M. Schwerhoff, and A. J. Summers, "Viper: A verification infrastructure for permission-based reasoning," 2014.

[61] "VeriFast License". Katholieke Universiteit Leuven. Accessed: Aug 18, 2019. [Online]. Available: https://github.com/verifast/verifast/blob/master/LICENSE.md

[62] K. Leino, "Dafny: An automatic program verifier for functional correctness," vol. 6355, 2010, pp. 348–370.

[63] J. W. McCormick, *Building High Integrity Applications with SPARK*, 2015.

[64] "Dafny: A Language and Program Verifier for Functional Correctness". Microsoft. Accessed: Aug 17, 2019. [Online]. Available: https://www.microsoft.com/en-us/research/project/dafny-a-language-and-program-verifier-for-functional-correctness/

[65] "Dafny FAQ". Microsoft. Accessed: Aug 17, 2019. [Online]. Available: https://github.com/dafny-lang/dafny/wiki/FAQ

[66] "Spec#". Microsoft. Accessed: Aug 18, 2019. [Online]. Available: https://www.microsoft.com/en-us/research/project/spec/

[67] "VCC: A Verifier for Concurrent C". Microsoft. Accessed:June 22, 2019. [Online]. Available: https://www.microsoft.com/en-us/research/project/vcc-a-verifier-for-concurrent-c/

[68] "VCC License". Microsoft. Accessed: Aug 18, 2019. [Online]. Available: https://github.com/microsoft/vcc/blob/master/LICENSE

[69] "Appraisal OCPP". OpenChargeAlliance.org. Accessed: June 22, 2019. [Online]. Available: www.openchargealliance.org/about-us/appraisal-ocpp/

[70] "OCA OCPP 2.0 Part 0 - Introduction". OpenChargeAlliance.org. Accessed: June 22, 2019. [Online]. Available: https://www.openchargealliance.org/downloads/

[71] "OCA OCPP 2.0 Part 1 - Architecture & Topology". OpenChargeAlliance.org. Accessed: June 22, 2019. [Online]. Available: https://www.openchargealliance.org/downloads/

[72] "OCA OCPP 2.0 Part 2 - Appendices". OpenChargeAlliance.org. Accessed: June 22, 2019. [Online]. Available: https://www.openchargealliance.org/downloads/

[73] "OCA OCPP 2.0 Part 2 - Specification". OpenChargeAlliance.org. Accessed: June 22, 2019. [Online]. Available: https://www.openchargealliance.org/downloads/

[74] "OCA OCPP 2.0 Part 4 - OCPP-J Specification". OpenChargeAlliance.org. Accessed: June 22, 2019. [Online]. Available: https://www.openchargealliance.org/downloads/

[75] "Does your program do what it is supposed to do?". OpenJML. Accessed: June 22, 2019. [Online]. Available: https://github.com/verifast/verifast/

[76] D. J. Pearce and L. Groves, "Whiley: A platform for research in software verification," in *Software Language Engineering*, M. Erwig, R. F. Paige, and E. Van Wyk, Eds. Cham: Springer International Publishing, 2013, pp. 238–248.

[77] P. Philippaerts, J. T. Mhlberg, W. Penninckx, J. Smans, B. Jacobs, and F. Piessens, "Software verification with verifast: Industrial case studies," *Science of Computer Programming*, vol. 82, no. C, pp. 77–97, 2014.

[78] T. Runge, I. Schaefer, L. Cleophas, T. Thüm, D. Kourie, and B. Watson, "Tool support for correctness-by-construction," in *Fundamental*

Approaches to Software Engineering - 22nd International Conference, FASE 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Proceedings, ser. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), R. Hähnle and W. van der Aalst, Eds. Germany: Springer, 1 2019, pp. 25–42.

[79] M. Utting, D. Pearce, and L. Groves, "Making Whiley Boogie!" vol. 10510. Springer Verlag, 2017, pp. 69–84.

[80] "Research prototype tool for modular formal verification of C and Java programs". VeriFast. Accessed: June 22, 2019. [Online]. Available: https://github.com/verifast/verifast/

[81] F. Vogels, B. Jacobs, and F. Piessens, "Featherweight Verifast," *Logical Methods in Computer Science*, vol. 11, no. 3:19, p. 157, 2015. [Online]. Available: https://lmcs.episciences.org/1595

[82] "Whiley A Programming Language with Extended Static Checking". Whiley. Accessed: Aug 18, 2019. [Online]. Available: http://whiley.org/

[83] "Why3 Where Programs Meet Provers". Why3. Accessed: Aug 18, 2019. [Online]. Available: http://why3.lri.fr/