COTS Hardware and Free Software Components for Safety Critical Systems in Developing Countries

D. W. Carr, R. Ruelas
Departamento de Ingeniería de Proyectos, Universidad de Guadalajara
Apdo. Postal 307, C.P. 45101 Zapopan, Jalisco, Mexico
doncarr@gmail.com, rruelas@newton.dip.udg.mx

Raul Aquino Santos
Facultad de Telematica, Universidad de Colima
Colima, Colima, Mexico.
aquinor@ucol.mx

Apolinar Gonzalez Potes
Facultad de Ingeniería de Mecanica y Electrica, Universidad de Colima
Colima, Colima, Mexico.
apogon@ucol.mx

Abstract

Traditionally, safety critical systems have been constructed from low volume hardware and software components specifically designed for safety critical systems. Almost all are closed systems, and the exact designs and source code are not available for analysis or comparison; we can only evaluate the reliability based on published reports of accidents attributed to the control systems. These systems typically cost into the tens of millions of dollars, and can easily run into the hundreds of millions of dollars, often putting them out of reach in developing countries, or taking funding away from other projects where more lives could be saved. It is the belief of the authors, that safety critical systems can be created from high volume general purpose COTS hardware and open source components and be just as reliable as traditional low volume hardware and closed source software components. We are in fact working with the light rail system in Guadalajara Mexico, and have an agreement to develop various projects, including a monitoring/signaling system using open source software licensed under the GNU General Public License (GPL).

1 Introduction

Prices for safety critical systems remain out of reach for many developing countries, or take money away from other underfunded programs where many lives are lost each year. We believe that systems constructed from COTS hardware and free software components, together with application modules written locally using N-Version programming techniques[1][2][3], internal consistency checks, world-wide peer review, and other techniques, can obtain a level of safety similar to that of a commercial system costing millions more. We do not have any way of verifying the level of safety of commercial systems, but in any case, given the number of lives that could be saved with the savings applied to road safety systems, improved traffic lights, better health care, etc, there is almost no question that we can save more lives with the same amount of money that would go for a commercial system. And, given that these kinds of real-time safety-critical systems developed and deployed locally can reduce dependence on foreign technology and know-how, and also serve as educational opportunities, the advantages of locally developed systems based on COTS and free software are many.

At the light rail system in Guadalajara, Mexico, we have implemented two GPL licensed systems, one to monitor the a pumping station and generate alarms, and another to monitor the departures at the four extremes of the current light rail system. The second should be classified as a safety critical system, since, operators will become dependent on it to maintain the spacing of trains and thus the safety of the system. Although we can not offer proof for all of the varied circumstances, we believe that in almost all developing countries, more lives can be saved by spending less on proprietary safety critical control systems, assuming that the savings are spent in other areas that are currently under funded such as health care, road safety, enforcement of speed limits, etc.

2 Costs

Obviously, costs for proprietary safety critical control systems can run into the tens or hundreds of millions of dollars. In one example that the authors are aware of, a new light rail signaling system upgrade was quoted at over 10 million dollars, when the cost of hardware was independently valued at less then 300,000 dollars. This means that the combination of hardware/software/installation is more than 9 million dollars. The CBTC system that is being tested on one line in New York City, has already cost into the hundreds of millions of dollars. The question we need to ask, is how much would these types of systems cost if we could create open source versions with all of the software and hardware design open and based on COTS hardware. It should be rather obvious that we can save at least on the order of millions for each application, up to tens of millions in some cases. For the light rail systems, the good news is that after the first application, there are not recurring software and hardware design costs, you only have to pay for the new hardware, and the engineering to apply everything to a new line.

3 Software Reliability

There has been a wealth of research in the area closed source versus open source security [7][8][12][9][10][11][13] that is also applicable to the question of whether open source or closed source is better for safety critical systems. This is maybe better know as the security by obscurity debate. This research differs in that almost all safety critical systems are kept on private networks and/or are physically isolated, and we are thus more focussed on the correctness of the system than whether or not it is crackable. We can not completely discount the possibility of a malicious person or persons gaining access to the safety critical system, but, this definitely changes the focus of the design where correctness of the application becomes much more critical than security. Also, if malicious persons were able to access the area of safety critical systems, it would most probably be much easier to sabotage other physical parts of the system, than to crack the software. Actually, according to Anderson [12], this should tip the balance toward open source systems, as the reliability gained by all interested parties in the world being able to study the source code, is not offset as much by malicious persons being able to use the source code to formulate attacks, since safety critical systems are on private networks and in any case, malicious persons with physical access would find it much easier to target other parts of the system than try to find vulnerabilities in the software.

Just as with commercial software where there has been a lot of foot dragging and cover-ups of software defects, we expect this to be a problem for safety critical systems as well. This is actually compounded with safety critical systems since the binaries are even private, such that it would be quite simple to fix safety critical bugs silently and privately. Commercial software companies have been caught doing just this, since the binaries are available and it is possible to see the changes. We expect that there have been many other incidents like this that have been rolled up in service packs, and never disclosed. But, the fact that this sort of thing can be done privately and silently with safety critical software at the very least deprives us of good knowledge about defects in proprietary safety critical systems. A vendor that was currently bidding on a large contract would be highly motivated to reduce any bad publicity for instance.

One additional point with GPL software, is that extensions to GPL licensed software can be kept private, as long as they are not distributed, so that critical parts of a security scheme could be kept private for individual customers. As an example, for a given system, we could use a custom message format and encryption methods to make it harder for crackers able to infiltrate the network and send false messages. The GPL version three actually extends these abilities, allowing extensions written by third party consultants to be kept private as well. Thus, a certain level of "obscurity" can be obtained even with GPL licensed software.

4 Software reliability of code specific to the control system that is not high volume free software

Obviously, the part of the code specific to the safety-critical control system will be low volume. With time, we expect that there will be widely used safety-critical control systems, but they will never achieve the level of use for other types of free software components. So, for these low volume pieces, we will use N-Version programming techniques [1][2][3], internal consistency checks, word-wide peer review, design patterns, avoid known dangerous coding practices and constructs, avoid known dangers system calls and library functions, and just good programming practices in general. We hope that the world-wide peer review will also greatly increase the safety of these low volume free software based components as researchers and other interested parties world wide review the code. We are hoping that Universities will review this code since, previously, there has been almost no safety critical software available for students and researchers to analyze. Even old decommissioned software systems are not made public and kept private, possibly for fear of being embarrassed by researchers on the outside from finding defects, and also fear of legal liability. In any case, the complete lack of public review in our humble opinion actually reduces the safety of systems and allows vendors to hide defects and silently repair safety-critical issues.

5 Hardware reliability

To achieve the level of hardware reliability required by safety critical applications there are a variety of vendors supplying COTS hardware that has been rigorously tested to destruction [14], and is high volume. This high volume of systems sold gives an advantage to COTS systems over custom low volume hardware that is not as extensively customer tested in as many diverse applications. The same can be said for high volume free software components that are not only tested by millions of customers world-wide in many varied conditions, but the actual source code can be reviewed without restriction by any interested parties world-wide.

6 Conclusions

Prices for safety critical systems remain out of reach for many developing countries, or take money away from other underfunded programs where many lives are lost each year. We believe that systems constructed from COTS hardware and free software components, together with application modules written locally using N-Version programming techniques, internal consistency checks, world-wide peer review, and other techniques, can obtain a level of safety similar to that of a commercial system costing possibly tens or hundreds of millions of dollars more. In any case, given the number of lives that could be saved with tens or hundreds of millions of dollars applied to road safety systems, improved traffic lights, better health care, etc, there is very good possibility that we can save more lives with the same amount of money that would go for a commercial system. Since these kinds of real-time safety-critical systems developed and deployed locally can reduce dependence on foreign technology and know-how, and also serve as educational opportunities, there are many other advantages for locally developed systems based on COTS hardware and free software components.

References

[1] A. Avizienis, 1995, *The Methodology of N-Version Programming*, SOFTWARE FAULT TOLERANCE, CH. 2, JOHN WILEY & SON INC, pp23–46

[2] J. Dugan and R. Lyu, 1994, *System Reliability Analysis of an N-version Programming Application*, IEEE TRANSACTIONS ON RELIABILITY, VOL. 43, NO. 4, pp513–519

[3] L. Hutton, 1997, *N-version design versus one good version*, IEEE SOFTWARE, NOVEMBER/DECEMBER 1997, pp71–76

[4] D. W. Carr, R. Ruelas, J. F. Gutierriez-Ramirez and H. Salcedo-Becerra, 2005, *A Communications Based Train Control System Using a Modified Method of N-Version Programming*, CONGRESO DE INSTRUMENTACIÓN (SOMI XX), León, Gto., Mexico, 2005.

[5] D. W. Carr, R. Ruelas, J. F. Gutierriez-Ramirez and H. Salcedo-Becerra, 2005, *An Open On-Board CBTC Controller based on N-Version Programming*, PROC. IEEE INTERNATIONAL CONFERENCE ON COMPUTATIONAL INTELLIGENCE FOR MODELLING CONTROL AND AUTOMATION (CIMCA'05), Vienna, Austria.

[6] H. Salcedo-Becerra, D. W. Carr, R. Ruelas, G. A. Ponce-Castaeda, 2006, *Performance Monitoring for the Light Rail System in Guadalajara, Mexico* INT. CONF. ON DYNAMICS, IN-

strumentation and Control, August 13-16, 2006, Querétaro, Mexico.

[7] Ross J. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, Wiley, ISBN: 978-0471389224, 640 pages

[8] Ross Anderson, *Security in open versus closed systems the dance of Boltzmann, Coase and Moore* Conference on Open Source Software Economics, Toulouse, France, 2002

[9] Jaap-Henk Hoepman, Bart Jacobs*Software Security Through Open Source* April, 2005, http://citeseer.ist.psu.edu/hoepman04software.html

[10] Eric S. Raymond,*The Cathedral & the Bazaar, Musings on Linux and Open Source by an Accidental Revolutionary*,O'Reilly & Assoc., 1999, http://www.oreilly.com/catalog/cb/

[11] Crispin Cowan, *Software Security for Open Source Systems*, IEEE Computer Society, IEEE Security & Privacy, 2003

[12] Ross Anderson, *Open and Closed Systems are Equivalent (that is, in an ideal world)*, http://citeseer.ist.psu.edu/668772.html

[13] Marit Hansen, Kristian Khntopp, Andreas Pfitzmann, *The Open Source Approach - Opportunities and Limitations with Respect to Security and Privacy*, Computers and Security, 2002, pp461–471

[14] Report on Highly Accelerated Life Test (HALT) for the Technologic Systems TS-7250 conducted from February 21st through 23rd, 2006, at QualMark Labs in Santa Clara, California