Science of Computer Programming 91 (2014) 139–140

Contents lists available at ScienceDirect

Science of Computer Programming

journal homepage: www.elsevier.com/locate/scico

Preface

Selected contributions from the Open Source Software Certification (OpenCert) workshops

We present to you this special issue dedicated to the 2nd, 3rd and 4th editions of the International Workshop on Foundations and Techniques for Open Source Software Certification (OpenCert) held in 2008 (Milan, Italy), 2009 (York, UK) and 2010 (Pisa, Italy) respectively. This is a compilation of a selected set of extended papers presented at these workshops.

OpenCert provides for a unique venue advancing the state of the art in the analysis and assurance of open source software with an ultimate aim of achieving certification and standards. The dramatic growth in open source software over recent years has provided for a fertile ground for fundamental research and demonstrative case studies. Over the years, OpenCert has enabled a thriving community, small but focused, examining issues ranging from certification to security and safety analysis for applications areas as diverse as railways, aviation, knowledge management, sustainable development, and the open source developers community.

Writing correct C programs is well-known to be hard, not least due to the many language features intrinsic to C. Writing secure C programs is even harder. This is precisely what Olesen et al. set out to address in their paper titled *"Coccinelle: Tool support for automated CERT C Secure Coding Standard certification"*. They aim to develop a prototype tool to ensure that C programs comply with US CERT standards for secure C coding. Coccinelle is a program matching and transformation engine providing for specifying desired matches and transformations in C code. Clang is a source code analyser for security. By combining the two, the authors aim to assist the task of program writing and easy compliance with secure coding standards.

Breuer and Pickin take advantage of the modern shift towards the cloud computing paradigm and propose a 'volunteer cloud' to analyse large open source code bases. In their paper titled *"Open Source Verification in an Anonymous Volunteer Network"* they show that the computation may be handled incrementally by the client CPUs of the distributed 'volunteer cloud' essentially each taking a fragment of the work at a time. An experiment to demonstrate this for a million lines of C code serves to validate their approach. This may very well be the future of a crowd-sourced verification.

Open approaches to software and models increasingly find their way to serve safety-critical systems, and railway signalling and control is certainly one such area. Feuser and Peleska address this challenge, in their contribution titled *"Dependability in Open Proof Software with Hardware Virtualization—The Railway Control Systems Perspective"*, by examining the openETCS initiative: this is an effort to bring the rigour of formal methods for specification and analysis to the European Train Control System (ETCS) standard. They particularly narrow down to ensure that the safety of a mixed open/closed source system is addressed by mechanisms to prevent software components of minor trustworthiness to corrupt the behaviour of a trusted safety-critical core. Lessons drawn from this work would be undoubtedly valuable in several other domains.

Almeida et al., in their paper titled *"CAOVerif: An open-source deductive verification platform for cryptographic software implementations"*, tackle a domain-specific language for cryptographic software, namely CAO, which is part of an open source toolbox dedicated to cryptographic algorithms and protocol implementation. As part of their work they present a model in first-order logic and use a deductive verification approach to verify CAO code for cryptography-relevant security properties.

With over 500 known distributions of Linux available, one challenge for software developers is to ensure portability of their applications across all distributions. Rubanov and Silakov address this problem in their paper titled *"Ensuring Portability of Linux Applications through Standardization and Knowledge Base Driven Analysis"*. They propose an approach to automatic portability for Linux applications, and in the context of a Linux Standard Base (LSB), which is an attempt to standardise common interfaces, they describe the Linux Application Checker tool to help explore external dependencies for applications.

Sowe et al. present an *"An Empirical Study of FOSS Developers Patterns of Contribution: Challenges for Data Linkage and Analysis"* where the open source developer community's contribution is analysed using Bayesian networks to look for patterns of contribution and the impact on end-product quality. This work serves to provide insights into various aspects of developer behaviour and how this could be used at an advantage in terms of tools and project leadership. Undoubtedly,

http://dx.doi.org/10.1016/j.scico.2014.04.007

such efforts will help realise the various critical factors underpinning open source software projects that are successful from those that are not.

We would like to express our gratitude to all the people who have made this special issue possible. First and foremost, our most sincere appreciation is to the authors for submitting their papers and incorporating all the corrections and improvements as advised by a thorough reviewing process. We are indebted to the reviewers for kindly contributing their time and effort to ensure the highest quality of each paper. Last but not least, we would like to thank Bas van Vlijmen and Jan Bergstra, and the editorial staff at Elsevier for agreeing to publish this special issue as a volume in Science of Computer Programming, and their assistance in bringing this special issue to publication.

Luís Soares Barbosa
University of Minho, Portugal

Siraj Ahmed Shaikh*
Coventry University, United Kingdom
E-mail address: s.shaikh@coventry.ac.uk.

8 April 2014
Available online 24 April 2014

* Corresponding editor.