Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 238 (2009) 11–17
www.elsevier.com/locate/entcs

A Software Certification Consortium and its Top 9 Hurdles

John Hatcliff[a,1], Mats Heimdahl[b,2], Mark Lawford[c,3], Tom Maibaum[c,3], Alan Wassyng[c,3,4], Fred Wurden[d,5]

[a] *Department of Computing and Information Sciences, Kansas State University, Manhattan, KS, USA*

[b] *U of Minnesota Software Engineering Center, Computer Science and Engineering, University of Minnesota, Minneapolis, MN, USA*

[c] *Software Quality Research Lab, McMaster University, Hamilton, ON, Canada*

[d] *Microsoft Corp., Seattle, WA, USA*

Abstract

In August of 2007 and December of 2007, North American academic researchers, industry representatives and regulators were invited to meetings in Washington and Minneapolis, respectively, with the goal of forming a Software Certification Consortium (SCC). At the first meeting, objectives were established for the consortium and a certification grand challenge was issued. At the second meeting, all participants were asked to complete the statement: "Software certification is hard because . . .". The group then synthesized the results into a "Top 9" list by means of discussion and voting. In this article, we describe the goals that we believe should be the goals of SCC, via details of these Top 9 hurdles that are preventing us from making software certification part of the mainstream.

Keywords: Software Certification Consortium (SCC), Objectives, Projects, Formal Methods

1 Introduction

People are increasingly dependent upon software in their daily lives. In addition to all the conventional software driven systems we are familiar with in our offices and homes, software in embedded systems implements the control algorithm in anti-lock brakes in our cars, fly-by-wire systems in airplanes, nuclear power plants, and life saving biomedical systems. All of these systems are "hard real-time" systems that must react with precise timing properties to function correctly. In an effort

[1] Email: hatcliff@cis.ksu.edu
[2] Email: heimdahl@cs.umn.edu
[3] This work was partially supported by NSERC
[4] Email: lawford,maibaum,wassyng@mcmaster.ca
[5] Email: fredwurd@microsoft.com

doi:10.1016/j.entcs.2009.09.002

to improve the quality of these real-time software systems and thereby reduce the risk to the public of system failure, extensive research has been carried out on formal, mathematical specification, verification and validation techniques. While these methods are beginning to show promise in improving software quality, it is not always clear to software practitioners how well these theoretical techniques model real embedded systems and how they can be applied to practical industrial systems. The jury is still out as to whether some of the formal techniques are really only useful on academic examples. On the other hand, it is also becoming clear to industry and regulators that conventional techniques based on conformance to development processes and on testing techniques, no matter how extensive, cannot guarantee system properties to a sufficiently satisfactory level.

One of the best indicators of this concern is the recently published National Academy of Sciences Report on Software for Dependable Systems [1]. This report, which we predict will prove very influential in guiding regulators and industry, documents the current state of the art in software development - and what needs to be done in the future.

The report emphasizes two *Findings*. The first of these is that software development has to improve in order to deliver more dependable software-based systems in a world in which software plays an ever-increasing role. The second is that we need to document and analyse software failures in order to understand the contributing factors, especially if the contributing factors were related to the development process. In addition, the report makes a number of important *Recommendations*. The recommendations are targeted at two distinct groups: i) Builders and Users of Software, and ii) Supporters of Software Education and Research. Recommendations to the first group include: use formal methods, software development technologies and principles that are known to be effective; build dependability cases that include security concerns; do not rely solely on process and testing to provide dependability; demand transparency and accountability; base certification on inspection and analysis of dependability claims and evidence. Recommendations to the second group include: place greater emphasis on dependability in the education of software professionals/researchers; fund basic research to improve dependability of systems that contain software, with an emphasis on evidence of dependability.

We agree, substantially, with the findings and recommendations of this report. We believe that it is implicit in the report, but should be stated more emphatically, that we need to use what we already know about building highly dependable software, and that we need to conduct more research on how to build *and certify* software-based systems, in which the focus should shift from process to product.

1.1 The Goal of Certification

The goal of certification is to systematically determine, based on the principles of science, engineering and measurement theory, whether an artifact satisfies accepted, well defined and measurable criteria.