Information and Software Technology 60 (2015) 102–112

Contents lists available at ScienceDirect

Information and Software Technology

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

An empirical research agenda for understanding formal methods productivity

Ross Jeffery *, Mark Staples, June Andronick, Gerwin Klein, Toby Murray

NICTA, Australia
University of New South Wales, Australia
223 Anzac Parade, Kensington, NSW 2052, Australia

ARTICLE INFO

Article history:
Received 23 June 2014
Received in revised form 13 November 2014
Accepted 15 November 2014
Available online 23 December 2014

Keywords:
Empirical software engineering
Productivity
GQM
Formal methods
Formal verification
Proof Engineering

ABSTRACT

Context: Formal methods, and particularly formal verification, is becoming more feasible to use in the engineering of large highly dependable software-based systems, but so far has had little rigorous empirical study. Its artefacts and activities are different to those of conventional software engineering, and the nature and drivers of productivity for formal methods are not yet understood.
Objective: To develop a research agenda for the empirical study of productivity in software projects using formal methods and in particular formal verification. To this end we aim to identify research questions about productivity in formal methods, and survey existing literature on these questions to establish face validity of these questions. And further we aim to identify metrics and data sources relevant to these questions.
Method: We define a space of GQM goals as an investigative framework, focusing on productivity from the perspective of managers of projects using formal methods. We then derive questions for these goals using Easterbrook et al.'s (2008) taxonomy of research questions. To establish face validity, we document the literature to date that reflects on these questions and then explore possible metrics related to these questions. Extensive use is made of literature concerning the L4.verified project completed within NICTA, as it is one of the few projects to achieve code-level formal verification for a large-scale industrially deployed software system.
Results: We identify more than thirty research questions on the topic in need of investigation. These questions arise not just out of the new type of project context, but also because of the different artefacts and activities in formal methods projects. Prior literature supports the need for research on the questions in our catalogue, but as yet provides little evidence about them. Metrics are identified that would be needed to investigate the questions. Thus although it is obvious that at the highest level concepts such as size, effort, rework and so on are common to all software projects, in the case of formal methods, measurement at the micro level for these concepts will exhibit significant differences.
Conclusions: Empirical software engineering for formal methods is a large open research field. For the empirical software engineering community our paper provides a view into the entities and research questions in this domain. For the formal methods community we identify some of the benefits that empirical studies could bring to the effective management of large formal methods projects, and list some basic metrics and data sources that could support empirical studies. Understanding productivity is important in its own right for efficient software engineering practice, but can also support future research on cost-effectiveness of formal methods, and on the emerging field of Proof Engineering.
© 2014 Elsevier B.V. All rights reserved.

1. Introduction

Formal methods is the mathematical specification, design and verification of computer systems. It can provide a much higher level of assurance than traditional code-and-test approaches to software engineering. Software engineering researchers have aspired to see the wide-spread use of formal methods since the 1970s, but only

* Corresponding author at: University of New South Wales, Australia. Tel.: +61 2 9376 2000.
E-mail address: ross.jeffery@nicta.com.au (R. Jeffery).

http://dx.doi.org/10.1016/j.infsof.2014.11.005
0950-5849/© 2014 Elsevier B.V. All rights reserved.

recently have technologies and techniques developed enough for it to become practical for use in non-trivial systems development projects. Increasingly, software systems are also safety- or security-critical and so could benefit from formal methods verification to provide direct evidence about system dependability. Nonetheless, to broaden its reach requires that its costs be better understood and where possible reduced. In particular, as noted by Klein [29], verification of low level implementations has been considered to be prohibitively expensive until recently. But this view is changing with the development of newer tools and methods. Studies such as those by King et al. [27] show the benefits of formal specification and verification. In this study they were able to show that for the SHOLIS system, the use of a Z proof of properties at the requirements and design levels was "substantially more efficient at finding faults than the most efficient testing phase". In addition they concluded that verification of SPARK Ada code was "more efficient at error detection than unit testing". However studies such as this are still rare and the insights provided into formal methods productivity partial and inconsistent. Thus we need a deeper understanding of productivity in this context.

Formal methods involves different kinds of development artefacts than traditional software engineering, and can provide qualitatively different kinds of assurance. As will be discussed in this paper, some of the traditional metrics used in empirical software engineering do not apply in projects using formal methods. New metrics will be needed. In a previous paper [1], we claimed that there is not yet a good understanding of what to measure in projects using formal methods. There is a need for research on metrics, cost models and estimation methods for such projects. This echoes much earlier statements [15,48], which indicates the lack of progress in this field over many years.

In this paper we define a space of research questions about the productivity of formal methods, and define a collection of metrics that bear on these research questions. For each question, evidence provided in the literature pertaining to that question is shown. Our paper provides a research agenda and a call to researchers in Empirical Software Engineering to study formal methods projects, and for researchers and practitioners in formal methods to collaborate on the opportunity provided by these empirical studies.

The questions in this paper are relevant whether we want to (a) reduce the cost of formal methods, (b) have it scale better, or (c) provide evidence to compare its cost-effectiveness with conventional software engineering. For any of these goals we need to be able to characterize the cost of formal methods and uncover appropriate metrics for this context. We can then use these metrics to develop an understanding of task size and effort drivers, and the empirical relationships between these and effort and schedule.

The existing literature is used in three ways in this paper. Firstly, we motivate the work using published papers showing the current lack of empirical evidence concerning cost, effort and quality of formal methods application in industry. Secondly we use literature to test the face validity of the research questions proposed, and finally we summarize literature that addresses guidelines and management issues concerning the application of formal methods. This finally provides a summary of the state of practice and a picture of the need for careful empirical research on the use and characteristics of formal methods in industry.

2. Background

It is significant for this work that formal methods may use different activities to produce different artefacts as part of a different lifecycle compared with traditional software engineering. An illustration of this is shown in a formal verification process lifecycle model in Fig. 1. In this model major artefact differences include the proofs themselves and the set of invariants of the system, which would not be present in conventional software engineering. The different activities that create and maintain those artefacts are also present in the process lifecycle.

Formal methods may target functional correctness, but this is not the only "quality" of software. Security properties and real-time performance may also be able to be supported by formal methods, but qualities such as maintainability or portability do not have accepted mathematical formalizations.

Many empirical studies of conventional software engineering have examined the impact of process or technology on defect density (the number of defects per code size). With conventional software engineering, defects may be able to be reduced, but no guarantees can be provided that all defects have been identified or eliminated. In contrast, formal methods offers the possibility to create software with zero defects. (Subject to the soundness of the logical reasoning used and the empirical validity of the models of specifications and machines.) This makes it qualitatively different to conventional software engineering. Klein [29] notes "formal methods held much promise in the 1970s but failed to deliver". However he further states "this situation has changed drastically" illustrating that formal verification of medium-level models of large systems of the order of 100,000 lines of C code has been demonstrated, and that formal verification of low-level implementations of systems around 10,000 lines of C code has also been achieved. The focus of Klein's paper [29] is on verification of operating systems, and it discusses in detail the Verisoft project and the L4.verified/seL4 project among others. What is revealed is that the methods, tools and skills available now make low-level verification a practical approach to be combined with testing and inspections in software development.

In their 2009 paper Woodcock et al. [50] find that "a weakness in the current situation is lack of a substantial body of technical and cost-benefit evidence from application of formal methods and verification technology". In summarizing their survey they report that 35% of respondents showed improvement in time and 12% a worsening. With regard to cost 37% reported an improvement and 7% a worsening, and with respect to quality, 92% reported an improvement and 8% reported worse quality. It is clear from this that a deeper empirical understanding is needed of the use of formal methods and their impact on cost, quality, and schedule. Woodcock et al. [50] also provide eight case study descriptions of relatively recent industrial projects using formal methods. These are then used to observe that formal methods have "not seen widespread adoption...except arguably in the development of critical systems in certain domains".

In this paper we argue that given this, it is now time for the empirical software engineering community to add to our understanding of verification processes, artefacts, and contexts and to provide the necessary evidence needed to further the industrial application of formal verification in a cost effective manner. A first step in this is a deeper understanding of formal methods productivity. We focus intentionally on the use of formal methods in industrial scale projects as it is in this context that productivity becomes an issue and is additionally susceptible to empirical study.

The background literature is examined below as it relates to the research questions identified. A thorough review of the literature reveals little empirical study of the processes involved in the use of formal methods and inconsistent and incomplete findings to date concerning productivity in this domain.

3. Research goals

Our investigation uses the Goal Question Metric (GQM) approach [2,37]. GQM is a framework for the specification of a

Download English Version:

https://daneshyari.com/en/article/549817

Download Persian Version:

https://daneshyari.com/article/549817

Daneshyari.com