Towards Posit & Prove Calculi for Requirements Engineering and Software Design

In Honour of the Memory of Professor Ole–Johan Dahl

Dines Bjørner

Computer Science and Engineering (CSE)
Informatics and Mathematical Modelling (IMM)
Building 322, Richard Petersens Plads
Technical University of Denmark (DTU)
DK–2800 Kgs.Lyngby, Denmark

db@imm.dtu.dk
http://www.imm.dtu.dk/~db/

Abstract. Some facts: Before software and computing systems can be developed, their requirements must be reasonably well understood. Before requirements can be finalised the application domain, as it is, must be fairly well understood. Some opinions: In today’s software and computing systems development very little, if anything is done, we claim, to establish fair understandings of the domain. It simply does not suffice, we further claim, to record assumptions about the domain when recording requirements. Far more radical theories of application domains must be at hand before requirements development is even attempted. In another (“earlier”) paper [6] we advocate(d) a strong rôle for domain engineering. We there argued that domain descriptions are far more stable than are requirements prescriptions for support of one or another set of domain activities. In the present paper we shall argue, that once, given extensive domain descriptions, it is comparatively faster to establish trustworthy and stable requirements than it is today. And we shall further, presently, argue that once we have a sufficient (varietal) collection of domain specific, ie. related, albeit distinct, requirements, we can develop far more reusable software components than using current approaches. In this contribution we shall thus reason, at a meta-level, about two major phases of software engineering: Requirements engineering, and software design. We shall suggest a number of requirements engineering and software design concerns, stages and steps

The paper represents work in progress. It is based on presentations of “topics for discussion” at the IFIP Working Group 2.3. Such presentations are necessarily of “work in progress” — with the aim of the presentation being to solicit comments. Hence the paper (“necessarily”) is not presenting “final” results.

O. Owe et al. (Eds.): From OO to FM (Dahl Festschrift), LNCS 2635, pp. 58–82, 2004.© Springer-Verlag Berlin Heidelberg 2004
1 Introduction

Our concern, in the present and in most of our work in the last almost 30 years, has been that of trying to come to grips with principles and techniques for software development.

The present paper sketches some such principles and techniques for some of the stages within the phases of requirements engineering and software design.

Our lecture notes, [7], the reader will find a rather comprehensive treatment of these and “most other related” software engineering issues!

1.1 Itemised Summary

Some facts:

– Before software and computing systems can be developed, their requirements must be reasonably well understood.
– Before requirements can be finalised the application domain, as it is, must be fairly well understood.

Some opinions:

– In today’s software and computing systems development very little, if anything is done, we claim, to establish fair understandings of the domain.
– It simply does not suffice, we further claim, to record assumptions about the domain when recording requirements.
– Far more radical theories of application domains must be at hand before requirements development is even attempted.

In another (“earlier”) paper [6] we advocate(d) a strong rôle for domain engineering.

– We there argued that domain descriptions are far more stable than are requirements prescriptions for support of one or another set of domain activities.
– In the present paper we shall argue, that once, given extensive domain descriptions, it is comparatively faster to establish trustworthy and stable requirements than it is today.
– And we shall further, presently, argue that once we have a sufficient (varietal) collection of domain specific, ie. related, albeit distinct, requirements, we can develop far more reusable software components than using current approaches.

In this contribution we shall thus reason, at a meta-level, about two major phases of software engineering:

– Requirements engineering, and
– software design.