Formal Methods for Early Fault Detection

Gerard J. Holzmann
gerard@research.bell-labs.com

Bell Laboratories, Murray Hill, NJ 07974, U.S.A.

Abstract. A traditional formal verification method becomes an effective weapon in the arsenal of a designer only after sufficient insight into a design problem has been developed for a draft solution to be formalized. In the initial phases of a design the designers can therefore perceive formal methods to be more of a hindrance than an assistance. Since formal methods are meant to be problem solving tools, we would like to find ways to make them both effective and attractive from the moment that a design process begins.

1 Introduction

Our aim is to develop a suite of tools that can support a design method based on the paradigm of early fault detection [4]. Tools of this type should be usable literally from the first day that a new design process starts, being applied to the early sketches of a new system. This means that these tools should be able to tolerate incomplete and sometimes inconsistent designs, and still be able to provide useful feedback to a designer.

Section 2 summarizes the design method based on early fault detection. Section 3 briefly discusses two light-weight design tools that we have built to support this paradigm, and comments on the first experiences with the application of these tools in an industrial design project. Section 4 discusses how the design decisions that are documented and analyzed with early fault detection tools can be used throughout a systems design cycle.

2 Scenario Based Design

The Devil’s Advocate. Formal methods have a number of recognizable characteristics. First, each method in this class is reported to be highly successful when it is applied by the developer of the method. Second, it is often used only reluctantly by traditional designers, and often on an experimental basis only. Third, each such experiment is considered to be a success by the initiators, but usually only a mixed blessing by the designers. Admittedly, all those participating in a formal methods experiment have reason to be content: the advocates, because we have one more documented case of a successful application, and the designers, because after the experiment is over, they are free to return to their regular mode of working. There is something not quite right about this, but what?
Too Much, Too Soon. The design of a new system usually starts in a rather tentative, exploratory, and iterative way with a Requirements Capture phase, as illustrated in Figure 1. The problem domain is surveyed, and fragments of a trial solution are sketched. Most of these sketches lead a short life, and are modified frequently. Some of them will survive and will become a permanent part of design documents, as soon as the understanding of the new system has settled sufficiently that such documents can indeed be written. In the initial phases of a design, all-out formal specification and verification techniques offer little help to the designer. They appear to require a level of formality and precision that is simply not available yet. The demands placed on the designer seem extreme: utter precision and detail is expected at a most inopportune time in the design cycle. In return, only fairly abstract properties may be established. The initial price to be paid is too high, the initial rewards are far too small.

The work on early fault detection methods attempts to bridge the gap between formal methods and design practice, by providing a more targeted set of tools for dealing with the early phases of a design. The full benefit of a formal method will first become evident during the high-level design phase, so all we need is a method that eases the transition to the more formal method, not one that makes absolute demands right from the start.

Instead of inventing a completely new design technique for the initial design phases, and then trying to persuade designers to adopt it, we have decided to study the informal techniques that designers naturally use, and consider if they could be sufficiently formalized to bridge the gap towards more formal methods in a natural way.

The Sketches Designers Make. Time sequence diagrams have been invented and reinvented by countless people. For better or for worse, they provide an irresistible informal technique for sketching sample behaviors of distributed sys-