EZ: A System for Automatic Prototyping of
Z Specifications

Veronika Doma and Robin Nicholl
Computer Science Department
The University of Western Ontario
London, Ontario
CANADA N6A 5B7

Abstract

Prototyping formal specifications can help capture accurately the requirements of real-world problems. We present a software system, called EZ, that generates executable prototypes directly from certain Z specifications. We first describe how nonmodular Z specifications can be mapped to search systems in C-Prolog. We then add modularity (schema referencing) and other features. A short comparison is made to other existing Z prototyping systems, with possibilities for future work being suggested.

1 Introduction

The goal of software engineering is to capture requirements of real-world problems in the form of programs that can be executed. For a particular problem, this process can involve several stages. In the earliest stage, or design process, a specifier may gather information about the problem and transfer this information to a non-procedural description, or specification, of the required input and output behaviour. An implementor may then convert the information in the specification to an executable program.

It is important that the implemented program be consistent with the specification, and that the specification be consistent with the real-world requirements. However, errors in understanding information and errors in encoding information are possible, and they can have serious ramifications. Testing of the implemented program helps to ensure that it is consistent with an implementor's understanding of a given specification. Also, proving correctness helps to ensure that the program is consistent with, or satisfies, its specification. These are methods for increasing confidence in the correctness of the implementation process. Prototyping a specification before the implementation process is begun can increase confidence in the design process.

Rather than refining a high-level description until it can be implemented efficiently, prototyping involves making a single mapping from a specification to a non-optimized implementation that exhibits the essential features of the specification. Viewing the execution of a prototype of an intended system can help a specifier to define and understand a real-word problem clearly, thus strengthening the correspondence between the true requirements of the desired system and its specification (see Figure 1). As does [3], we call the process of displaying the execution of a prototype animation.

1 authors' email addresses: veru@csd.uwo.ca, robin@csd.uwo.ca
Figure 1: Increasing the accuracy of transfer of user requirements to concrete specification: Animation of prototypes generated from a specification can help capture accurately the requirements of a real-world problem by giving feedback about the properties of the specification.

Since specifications can be easier and less costly to modify than implemented systems, prototyping a specification is also likely to save development time. This is especially true if the prototypes are created automatically via some consistent mechanical translation method. If the “distance” between the non-procedural specification language and the executable prototyping language is not very large, and if the prototype is not required to perform efficiently, then automatic prototyping from a formal specification language is feasible. Unfortunately, the expressiveness of many specification languages makes them undecidable, making it impossible to determine whether any particular animation will terminate. Nevertheless, prototypes that do terminate can provide useful information.

We chose the name EZ (for executin9 Z) for our prototyping system. Z has two powerful features which make it concise and intuitive to use. The first of these we call modularity in this paper, which allows the organization of components of a specification into chunks, or schemas, and allows referencing of these schemas from other schemas. The second of these is called the schema calculus, which allows expressions to be formed over schemas involving logical and compositional operators. Section 4 shows a method of implementing modularity and operators of the schema calculus, as well as other important features of Z, by first mapping Z specifications to search systems in Prolog and then extending these search systems with features that Prolog provides. Section 5 describes an implementation of the EZ system based on these findings.

There has been previous success in animating Z with Prolog. In the Alvey Project SuZan [3, 10, 8, 9], Z is translated into Prolog semi-automatically and some specifications can be executed based on an extensive library of Prolog primitives that have been implemented. A generate-and-test method is used to generate variable values and test constraints over them. The direct execution of the resulting Prolog is very slow and

2This term is not to be confused with that of modularity in [13], which proposes some (more) modular extensions to Z.