Experiences with a Specification Environment

Jacques Loeckx and Jörg Zeyer
University of Saarbrücken

Abstract. The OBSOCURE system is an environment intended to support the design and verification of specifications. After a short informal description of the system the paper sketches the case studies carried out in the framework of the KORSO project. Based on the experience gained and on a comparison with other systems used within the KORSO project, the paper critically discusses the use of specifications in the design of correct software in general and the features of the OBSOCURE system in particular.

1 Introduction

In the last decade a large number of powerful specification methods and specification languages for abstract data types have been presented in the literature. Unfortunately, in most cases their use has been illustrated in toy examples only. An important goal of the KORSO project was therefore to show the practicability of specifications in "real-life" software development. To this end it was decided to carry through several "medium-sized" case studies. The OBSOCURE system was chosen as one of the tools to achieve this goal.

The present paper reports upon the treatment of three medium-sized case studies carried through at Saarbrücken with the help of the OBSOCURE system. It discusses the differences with the SPECTRUM specifications for the same case studies drawn up "by hand" by the KORSO partners of Munich and Berlin. Finally, the paper critically analyses the use of specifications in the design of correct software in general and discusses the features of the tools used in particular.

Section 2 presents the tool used in Saarbrücken for the treatment of the case studies. This tool — called the OBSOCURE system — constitutes a specification environment for the specification language OBSOCURE. The three case studies treated with this environment are shortly described in Section 3. Finally, Section 4 contains a critical discussion on specification and thus constitutes the essential part of the paper.

2 The OBSOCURE system

Writing a specification not exceeding one page is possible by using pencil and paper. Writing larger specifications soon appears to be very tedious for at least two reasons: First, one is lead to introduce a large number of sorts and operations and it may be difficult to keep all of them in mind. More importantly, when specifications are put together they generally strongly "interfere". Controlling
this interference requires checking more or less elaborate context conditions and verifying properties that guarantee persistency. Consequently, writing non-trivial specifications requires the use of an environment supporting the design and the verification of specifications.

Even a syntactically correct and semantically consistent specification may fail to be adequate, i.e. may fail to capture the problem to be specified (cf. [Cho93]). Being a relation between an informal notion (viz. a problem) and a formal one (viz. a specification) adequacy cannot be proved but merely be tested. Rapid prototyping is a classical way to perform such adequacy tests. An environment should therefore also contain tools for rapid prototyping.

The OBSCURE system constitutes such an environment. It is based on the use of the specification language OBSCURE or, more precisely, on a particular instantiation of this language called OBSCURE-S. Section 2.1 and 2.2 shortly describe those features of these two specification languages that are needed for the understanding of the rest of the paper. A description of the OBSCURE system itself is in Section 2.3 and 2.4.

2.1 The specification language OBSCURE

Specification languages such as CLEAR [San84], PLUSS [BGM89] or ASL [SW83] aim at generality and insist on semantical issues. Instead, the specification language OBSCURE was designed in view of practical use. It therefore concentrates on simplicity and handiness. In particular, it imposes stringent context conditions that allow to keep the semantics transparent. More recent specification languages such as SPECTRUM [BFG93] contain more powerful constructs than OBSCURE allowing, for instance, polymorphism and higher-order operations. The price paid for these features is a more complex semantics.

A description of the specification language OBSCURE is outside the scope of the present paper. The reader may find a complete formal description in [LL93]. The now following remarks are intended to make the present paper self-contained.

As any other specification language OBSCURE provides constructs for putting specifications together and for renaming and hiding sorts and operations. In addition, OBSCURE provides the following three constructs. The construct axioms allows to restrict the class of models and to express, in particular, parameter constraints. The constructs subset and quotient allow to build subalgebras and quotient algebras respectively.

In OBSCURE two signatures are assigned to each specification. Informally speaking the import signature consists of those sorts and operations that are defined "outside" of the specification. The export signature consists of (a subset of) the import signature together with the sorts and operations defined "within" the specification. The goal of this distinction is essentially to enhance the modular structure of the specifications.

The application of a language construct may lead to a violation of the persistency or, more precisely, to an unintended restriction of the class of algebras specified. It may, in particular, restrict the class of algebras allowed as "import