Formal Aspects of Computing (2003) 3: 1–000

Response to "The Formal Specification of Safety Requirements for Storing Explosives"

Peter Gorm Larsen

The Institute of Applied Computer Science, Odense, Denmark

Abstract. This short communication is a response to [MS93] investigating their ACS system specification. The main point in this paper is that executing specifications can be used as a feasible way of validating them. It is essential to have tool support which enables one to write a generally not executable specification, and then prototype (parts of) it directly in the specification language, without translating it into some other prototyping language.

1. Introduction

The emphasis in [MS93] on following the guidelines from the interim MOD defence standard 00-55 is very interesting. However, we would like to focus on the need for appropriate tool support to gain the full benefit from using formal specifications. In particular we would like to stress that we feel that for industrial take-up of such formal techniques it is important not to use too many different notations in a project. It is not reasonable to expect that a software engineer is an expert in VDM-SL, OBJ3 and some implementation language like it was done in [MS93].

We used the IFAD VDM-SL Toolbox [Las93] to syntax and type-check the full ASCII version of the ACS specification[1]. The syntax of the VDM-SL proto-standard [ISO93] had changed in a few places and that was corrected. No errors were discovered here, but naturally this is also because the SpecBox [BFM89] tool had already been used for processing this specification. However, approximately 20 errors were discovered by the static semantics checker. These were minor

Correspondence and offprint requests to: Peter Gorm Larsen, IFAD, Forskerparken 10, DK – 5230 Odense M, Denmark.

[1] The authors of [MS93] were kind to let us have a copy of the full specification.

problems, but they would still yield problems in the implementation later on. We also feel reasonably sure that we would be able to discover more errors by prototyping the specification directly in VDM-SL (see more about this in the next section). However, we did not have time to do any testing in this way. Probably a tool supporting proofs[2] in VDM-SL would be able to discover minor problems in the proofs which have been carried out as well.

The major point here is that if we wish to gain the full benefit from using formal definitions we need to develop and use powerful tools supporting them.

2. Prototyping of specifications

The question about whether one should use executable specifications is closely related to the possible abstractness of the definitions. The specifications in [MS93] are mainly implicitly defined functions and operations. However, all of these definitions with one exception (*find-point*) can just as well be formulated explicitly without any loss of abstraction. This can be illustrated by a small example. The operation *ADD-OBJECT* is originally defined like:

```
1.0  ADD-OBJECT (o : Object-desc, obj : Object-label, site : Site-label)
 .1  ext wr pes : Site-label ⟷^m Pot-explosion-site
 .2  pre  site ∈ dom pes ∧
 .3       ∃ pt : Point · (safe-addition (o, pes (site).mgzn, pt) ∧
 .4                       obj ∉ dom pes (site).mgzn.objects)
 .5  post let p = pes~ (site) in
 .6       let mk-Point (x, y) = find-point (o, p.mgzn) in
 .7       let new-objs = p.mgzn.objects†{obj ↦ mk-Object (o, x, y)} in
 .8       let new-mag = μ (p.mgzn, objects ↦ new-objs) in
 .9       let new-site = μ (p, mgzn ↦ new-mag) in
.10       pes = pes~ † {site ↦ new-site}
```

It can alternatively be formulated (almost identically) explicitly by:

```
2.0  ADD-OBJECT : Object-desc × Object-label × Site-label →^o ()
 .1  ADD-OBJECT (o, obj, site) ≜
 .2    (def p = pes (site) in
 .3     let mk-Point (x, y) = find-point (o, p.mgzn) in
 .4     let new-objs = p.mgzn.objects † {obj ↦ mk-Object (o, x, y)} in
 .5     let new-mag = μ (p.mgzn, objects ↦ new-objs) in
 .6     let new-site = μ (p, mgzn ↦ new-mag) in
 .7     pes := pes † {site ↦ new-site})
 .8  pre  site ∈ dom pes ∧
 .9       ∃ pt : Point · (safe-addition (o, pes (site).mgzn, pt) ∧
.10                       obj ∉ dom pes (site).mgzn.objects)
```

Having looked at a number of specifications which use the pre-post style we claim that this is often the case (in many cases the result identifier is even directly set equal to some expression which uses the input parameters and input

[2] The IFAD VDM-SL Toolbox has currently no support for reasoning about specifications.

states). The reason for this may either be that users of VDM-SL use too low a level of abstraction or that the expressiveness of the explicit constructs make it possible to use a reasonable high level of abstraction.

Naturally we agree with [HJ89] that in general executable specifications can be misused and may get the user to think too operationally[3]. However, we believe (also stated in [AELL92]) that used with care[4] one can get a lot of insight in the correctness of a specification by executing the specification with a test suite.

3. Conclusion

We feel that [MS93] to some extent demonstrates that formal methods are mature enough to be used effectively in "real life" safety critical systems. This was done by detecting a number of contradictions and omissions in the informal requirement specification. It is very important to be able to reveal such problems because they can be very costly to reveal at a later stage. However, we also feel that [MS93] showed that tool support would be able to discover obvious problems with the formal specification. We think that in order to apply such techniques industrially it is essential that tool support can assist the usage of the formal specifications.

Concerning the prototyping of the specification we also think that it is essential it is done directly in the specification language, and not transformed into yet another language like OBJ3. The authors of [MS93] have informed us that this was a pragmatic rather than an ideological choice, based on existing experience and availability of tools. However, as pointed out in the paper another obstacle for the prototyping was MOD's refusal to provide test data. Naturally one would only gain the maximum benefit of the prototyping of the specification if one made an entire test suite and closely investigated whether the results are as expected. If the same test suite then can be used to test (validate) the corresponding implementation and it produces the same results one can gain even more confidence in the correctness of the implemented system.

Our conclusion is that formal methods currently are mature enough to be used for industrial development of "real life" software systems. We think that by means of training and appropriate tool support for formal methods it will be cost-effective to use such techniques today. We think that testing specifications presents an interesting alternative to fully formal development where all proof obligations are discharged. The man-effort, tool support and expertise needed to carry out a fully formal development is out of scope for most industrial projects. However, since testing is a well established discipline in industrial software development we believe that this could be a good place to start. One can then gradually use such a strategy in conjunction with proofs of essential properties.

---

[3] Writing implicit definitions do not "guarantee" a high level of abstraction. Implicit definitions can be misused as much as executable specifications with respect to the level of abstraction of the specification.

[4] It is essential that specifications are not written solely in order to execute them. The primary aim must always be to document *what* (and not too much *how*) a system shall do.

References

[AELL92] M. Andersen, R. Elmstrøm, P.B. Lassen, and P.G. Larsen. Making Specifications Executable – Using IPTES Meta-IV. *Microprocessing and Microprogramming*, 35(1-5):521–528, September 1992.

[BFM89] R. Bloomfield, P. Froome, and B. Monahan. SpecBox: A toolkit for BSI-VDM. *SafetyNet*, (5):4–7, 1989.

[HJ89] I.J. Hayes and C.B. Jones. Specifications are not (necessarily) executable. *Software Engineering Journal*, pages 330–338, November 1989.

[ISO93] Information Technology Programming Languages – VDM-SL. Technical report, First Committee Draft Standard: CD 13817-1, November 1993. ISO/IEC JTC1/SC22/WG19 N-20.

[Las93] P.B. Lassen. IFAD VDM-SL Toolbox. In J.C.P. Woodcock and P.G. Larsen, editors, *FME'93: Industrial-Strength Formal Methods*, page 681, Berlin Heidelberg, April 1993. Springer-Verlag.

[MS93] P. Mukherjee and V. Stavridou. The Formal Specification of Safety Requirements for Storing Explosives. *Formal Aspects of Computing*, 5(4):299–336, 1993.