Verifying OCL specifications of UML models : tool support and compositionality

Kyas, M.

Stellingen
behorende bij het proefschrift

Verifying OCL Specifications of UML Models: Tool Support and Compositionality

door
Marcel Kyas

I

When specifying systems one has to be aware of the subtle differences between *null* and *undefined*: Any programmer expects that *null* = *null* is true and that *undefined* = *undefined* is nonsense.

II

OCL cannot be used to specify the behaviour of operations, because: (i) the specification may call operations defined in the model as long as they are side-effect free, (ii) these operations can be overridden, even if they are defined in the OCL standard library, and (iii) *virtual binding* is used to resolve such calls. As a consequence, the meaning of constraints in a class diagram depends on its implementation.

III

Lamport and Paulson hold the opinion that mathematicians are so intelligent that their specification languages do not need to be typed [LP99]. Specification languages like OCL demonstrate the contrary.

IV

Karl Popper's remark that "whenever a theory appears to you as the only possible one, take this as a sign that you have neither understood the theory nor the problem which it was intended to solve" [Pop72] holds especially for UML.

V

UML 2.0 state machines can be rigorously formalised in about ten pages of rewriting logic [Sch05], which expose all ambiguities and unclarities [FSKdR05] occurring in the 68 page description in UML 2.0 [Obj04].

VI

UML state machines improve drastically on most *modern* object-oriented programming languages, whose semantics is based on ALGOL-60, by basing their semantics on Hewitt's actor model [Hew76].

VII

Some of the problems of proving industrial applications correct are: (i) The given specification is almost never correct. (ii) The given application is not structurally described, i.e., by composing simpler constructs to complicated ones in a hierarchical manner, also called by stepwise hierarchical refinement.

VIII

Completeness results are only relevant if the proof of completeness shows a generally applicable method for *de facto* constructing a proof for a correct program.

IX

Old-Norse poetry like *Ynglingatal* [AM 45 fol.] has been recorded by Christians. We have to mind this fact when arguing whether there was *sacral kingship* (where a king is viewed as a mediator or executive agent of a god) in pre-Christian Scandinavia of which features have been maintained by kings of Christian medieval

Scandinavia [Bae64]. We must also not forget that *our* reception of these poems is heavily influenced by our own culture [Fro51], which is strongly affected by Christianity.

X

The main problem of designing a distributed version of a Linda-tuple-space is not that Linda is inherently inefficient, but that it is difficult to find *reasonable* fairness requirements [Der05, Hlu05].

XI

Paul Lorenzen devised game semantics *(Dialogische Logik)*, because *every* scientist, especially humanists, should be able to reason formally [KL96]. However, most non-logicians do not apprehend game semantics.

XII

If inventions can be patented that do not make causally determined use of natural matter and energy, as is the case with software, then *all* teaching concerning mental activity becomes susceptible to patent litigation.

References

[AM 45 fol.] Am 45 fol. Codex Frisianus. Arnamagnæan Collection. Copenhagen, Denmark, ca. 1300–1325.

[Bae64] Walter Baetke. Yngvi und die Ynglinger. Eine quellenkritische Untersuchung über das nordische "Sakralkönigtum". *Sitzungsberichte der Sächsischen Akademie der Wissenschaften zu Leipzig*, 109(3), 1964.

[Der05] Alexander Derenbach. Client/Server-Architektur und Servertopologien eines verteilten Linda-Tupelraum in Java. Bachelor Thesis, Christian-Albrechts-Universität zu Kiel, October 2005.

[Fro51] Erich Fromm. *The Forgotten Language: An Introduction to the Understanding of Dreams, Fairy Tales and Myths*. Rinehart and Co., 1951.

[FSKdR05] Harald Fecher, Jens Schönborn, Marcel Kyas, and Willem-Paul de Roever. 29 new unclarities in the semantics of UML 2.0 state machines. In Kung-Kiu Lau and Richard Banach, editors, *Formal Methods and Software Engineering (ICFEM 2005)*, volume 3785 of *Lecture Notes in Computer Science*, pages 52–65. Springer-Verlag, 2005.

[Hew76] Carl Hewitt. Viewing control structures as patterns of passing messages. Technical Report 410, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, December 1976.

[Hlu05] Christopher Hlubek. Eine verteilte Tupelraum Implementierung in Java. Bachelor Thesis, Christian-Albrechts-Universität zu Kiel, October 2005.

[KL96] Wilhelm Kamlah and Paul Lorenzen. *Logische Propädeutik: Vorschule des vernünftigen Redens*. J.B. Metzler, Stuttgart, Weimar, 3rd edition, 1996.

[LP99] Leslie Lamport and Lawrence C. Paulson. Should your specification language be typed? *ACM Transactions on Programming Languages and Systems*, 21(3):502–526, May 1999.

[Obj04] Object Management Group. *UML 2.0 Superstructure Specification*, October 2004. `http://www.omg.org/cgi-bin/doc?ptc/2004-10-02`.

[Pop72] Karl Raymond Popper. *Objective Knowledge: An Evolutionary Approach*. Oxford University Press, 1972.

[Sch05] Jens Schönborn. Formal semantics of UML 2.0 behavioral state machines. Diploma Thesis, Christian-Albrechts-Universität zu Kiel, April 2005.