A CONCEPT OF PARADIGM IN THE MULTI-PARADIGM SOFTWARE DEVELOPMENT

Valentino Vranić

Department of Computer Science and Engineering, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology, Ilkovičova 3, 812 19 Bratislava, Slovak Republic

`vranic@elf.stuba.sk`

Several new software development paradigms (or programming paradigms, as they are often called), such as aspect-oriented programming, subject-oriented programming, and generative programming appeared recently as a reaction to still unsatisfactorily solved problems of the software engineering (e.g. reuse). Although these paradigms already carry the multi-paradigm flavor with them as they are based on other paradigms (e.g. aspect-oriented programming builds upon general procedure paradigms [3], generative programming involves aspect-oriented programming among other approaches [2]), the multi-paradigm approach definitely deserves to be taken special and explicit care of.

Multi-paradigm approach in programming is not a new idea. Unfortunately, it has been suppressed by the belief in the existence of "the best" programming paradigm that could solve some painful problems software engineering deals with. The main idea of the multi-paradigm software development is to apply several paradigms to a single software system (or a family of systems) development in a coexistent way [5].

The first step towards this goal is to define what is a software development paradigm. A paradigm in science and engineering disciplines represents a consistent collection of methods and techniques accepted by the relevant scientific community as a prevailing methodology of the specific field [5]. This general definition applies to software development paradigms as well, but it is not sufficient. In software development, paradigms are present at least at two different levels:

- large-scale paradigms [1], e.g. procedural paradigm, functional paradigm (discussion on various paradigms can be found in [4]), and
- small-scale paradigms, i.e. any commonality-variability pairing, e.g. overloading in C++: procedures with a common name with variability in algorithm [1].

It is clear that the definition introduced applies to the large-scale paradigms, which are appropriate when speaking in the context of software development process as a whole. The small-scale paradigms are language dependent and they are related to the meaning of the word *paradigm* as used in linguistics.

It has to be pointed out that, although programming languages are often designated as single-paradigm, considering the small-scale paradigms concept, they are multi-paradigm. A popular example is C++, which is often considered as an object-oriented language, although it supports several other paradigms (e.g. the procedural paradigm). Another example is Lisp, which is known as a functional programming language, but it supports imperative paradigm as well. Consider the assignment, characteristic for the imperative paradigm, that can be performed in Lisp like `(SET 'X 'Y)`.

The small-scale paradigms seem to be less elusive concept than the large-scale ones, but both are important and require to be explored in detail and formalized in order to enable their application through all the phases of the software development in a controlled manner. This task is to be performed as a future work.

This work was partially supported by Slovak Science Grant Agency, grant No. G1/7611/20.

References:

[1] J. O. Coplien: Multi-paradigm design and implementation in C++. In *Proc. of the GCSE '99 (co-hosted with the STJA 99)*, Erfurt, Germany, 1999. Presentation slides and notes, published on CD, available at http://www.bell-labs.com/~cope.

[2] K. Czarnecki: *Generative Programming: Principles and Techniques of Software Engineering Based on Automated Configuration and Fragment-Based Component Models*. Ph.D. thesis, Ilmenau Technical University, Germany, 1998. See http://www.prakinf.tu-ilmenau.de/~czarn.

[3] G. Kiczales et al.: Aspect-oriented programming. In M. Aksit and S. Matsuoka, editors, *Proc. of ECOOP'97—Object-Oriented Programming, 11th European Conference*, Jyväskylä, Finland, June 1997. Springer-Verlag LNCS 1241. Available at http://www.parc.xerox.com/aop.

[4] P. Návrat. A closer look at programming expertise: Critical survey of some methodological issues. *Information and Software Technology*, 38(1):37–46, 1996.

[5] V. Vranić: Multiple software development paradigms and multi-paradigm software development. In J. Zendulka, editor, *Proc. of ISM 2000 (part of MOSIS 2000)*, pages 191–196, Rožnov pod Radhoštěm, Czech Republic, May 2000. MARQ.