Domain-Specific Languages and Language Workbenches: The Future of Mainstream Programming

(Invited paper)

Igor Dejanovic
University of Novi Sad, Faculty of Technical Sciences
Novi Sad, Serbia
igord@uns.ac.rs

Abstract— **Software has become an indispensable part of our daily lives. It runs on our desktop and laptop computers, our smart phones, home appliances, our vehicles. There is an ever-rising pressure to reduce time to market for new software products and services. But despite the growing need for faster software development the way we make software has not significantly changed for the past few decades. The mainstream programming is still based on General Purpose Languages(GPL) of third generation. Domain Specific Language (DSL), in contrast to General Purpose Language, is a language particularly suited for well defined domain. By constraining language to a specific domain we can use concepts and constructs from the domain at hand. This alignment with the domain raises the level of abstraction and may increase productivity by the factor of 10 or more. Furthermore, DSLs enables better code analysis, verification and validation enabling creation of more robust end products. Notwithstanding the apparent boost in productivity and quality, widespread usage of DSLs is yet to be seen. Current state may be attributed to the fact that using DSLs involves language development and evolution which is an activity that can be very hard and costly if done without sophisticated tools. The development of DSLs is undertaken by a small community of developers. It is not uncommon to see DSLs developed to be used by only one institution, sometimes even on one project only. This is not an issue for GPL communities since GPLs evolve at slower pace and they are used by a larger community so the language tooling and know-how is accumulated over time. Additionally, being the mainstream technology the investment in the tools development is easier to justify. To facilitate the development and evolution of DSLs a robust Integrated Development Environments (IDEs) for this purpose have to be available to the developer's community. Those IDEs should support building of DSL infrastructure from language descriptions. Building editors, validators, debuggers, code generators, model/program transformation are some of the features that should be available in modern DSL IDE. Version control for both language and models/programs at the granularity of language concepts should be supported as well as a support for language/programs coevolution, i.e. semi-automatic migration of models/programs from old to the new language versions. This kind of integrated development environments would significantly cut down DSLs development and maintenance costs and thus made DSLs available to the wider circle of practitioners. This kind of IDEs for DSLs are nowadays generally known by the name of Language Workbenches, a term coined by Martin Fowler. This talk gives an overview of the current trends and state of the art tools for DSL development, presents our current research platform and gives our view of the future of programming.**