Reflexivity and Meta-Tools to Manage Your Software Real Estate

Darius Blasband

Numerous techniques are available to develop software while maintaining higher-level information about its architecture, limitations, features, dependencies, etc. However, in many cases, the only reliable information we have about a system's exact behaviour is made of large amounts of source code. The design documents, if any, have not been maintained up to date.

This tutorial describes tools and techniques for extracting information from large software code bases, and possibly performing large scale transformations on it. Topics covered by this tutorial typically include:

- Reverse engineering, abstracting design information from the code
- Metrics for portability, maintainability, etc.
- Checking how effectively tests exercise the code
- Impact analysis, measuring the effect of a planned change on the existing software base.
- Automated translations, between languages and libraries
- Checking compliance to industry or project guidelines
- Refactorings, that translate high-level design changes into existing code
- Componentization, where legacy code is integrated in modern technical frameworks
- Storage of structured parse trees through translation to canonic XML data structures

Tools and techniques used cover a range from Perl to the use of reflexive language features. The emphasis will be put on how does Object-Oriented technology contribute to these techniques, and even more importantly, how do these techniques apply to Object-Oriented code.

Darius Blasband was born in Brussels in 1965. He holds a master's degree and a PhD from the Université Libre de Bruxelles. He designed the YAFL object-oriented programming language in 1990, which introduced the concept of reusable and extensible compiler components to ease the design of domain-specific languages. He is also the main architect of RainCode, a software analysis and renovation line of products.

His interests are mainly focused on language design, type systems, compilation, automatic program understanding, and reverse engineering.