Towards More Safe Programming Language Constructs

Áron Baráth and Zoltán Porkoláb

Department of Programming Languages and Compilers,
Faculty of Informatics
Eötvös Loránd University

{baratharon|gsd}@caesar.elte.hu

Most of the current programming languages inherit their syntax and semantics from technology from the 20th century. Due to the backward compatibility, these properties are still unchanged, however newer technologies require different language constructs and different semantics. Instead of redefining the programming language, the developers patch the language with new library functions, or they add some –occasionally ambiguous– elements to the syntax [1]. Some languages provide very loose syntax, which is harmful, because it leads to critical errors [2, 3]. In other case the interleaving "normal" code and exception handling code can obfuscate the developer itself and the subsequent developers.

This paper presents several aspects of language elements such as basic and potentially unsafe elements of the syntax, control flow constructs, elements used in const-correctness, type-system, elements of multiparadigm programming –generative and functional–, capabilities of embedding a DSL, parallelism support, and taking account of branch prediction. These aspects determine the usablity, safety and learnability of a language. This paper also gives recommendation for a new and safe experimental programming language.

References

[1] G. Bracha, M. Odersky, D. Stoutamire, P. Wadler. Making the Future Safe for the Past: Adding Genericity to the Java Programming Language. In *OOPSLA '98 Proceedings of the 13th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, Pages 183–200, ACM New York, NY, USA, 1998

[2] https://gotofail.com

[3] http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-1266