Transforming client-server based Erlang programs to Erlang OTP design

Roland Király,
EKF - Institute of Mathematics and Informatics -
Department of Information Technology, Eger

NetWorkshop 2007

Abstract

Erlang/OTP is a functional programming environment designed for building concurrent and distributed fault-tolerant systems with soft real-time characteristics (like telecommunication systems). The core Erlang language consists of simple functional constructs extended with message passing to handle concurrency, and OTP is a set of design principles and libraries that supports building fault-tolerant systems. The language has a very strong dynamic nature that partly comes from concurrency and partly from dynamic language features.

Refactoring is a programming technique for improving the design of a program without changing its behaviour. In other words, you clean up your code but do not change what it does. Refactoring may precede a program modification or extension, preparing the program for the modification, or may be used after finishing the work in order to bring the program into a nicer shape. The transformations of refactoring can be used for optimisation as well.

The Erlang/OTP (Open Telecom Platform) is a porgramming language used by Ericsson, which is suitable for making concurrent and paralell programs. Programs created in this system are very fault-tolerant softwares and can be used efficiently to develop telecommunication systems.

The OPT extension of the language contains the gen-server, gen-fsm, gen-event and supervisor modules, which are formalizations of this common pattern. These behaviours can be used in the callback moduls which can export a pre-defined set of functions, the callback functions.

Erlang processes can not use a shared memory, so message passing is very important in handling distribution.

In my lecture I am going to present the way we can create refactoring steps to make transformations on client-server based programs to OTP design. I am also going to talk about the analysis of the references of function and variable scoping, about its analising techniques, and about the case studies of the implementation.

References

[1] Martin Fowler's refactoring site. http://www.refactoring.com/.

[2] H. Li, C. Reinke, and S. Thompson. Tool support for refactoring functional programs. *Haskell Workshop: Proceedings of the ACM SIGPLAN workshop on Haskell, Uppsala, Sweden*, pages 27–38, 2003.

[3] R. Szabó-Nacsa, P. Diviánszky, and Z. Horváth. Prototype environment for refactoring Clean programs. In *The Fourth Conference of PhD Students in Computer Science (CSCS 2004), Szeged, Hungary, July 1–4*, 2004.

[4] P. Diviánszky, R. Szabó-Nacsa, and Z. Horváth. Refactoring via database representation.

[5] J. Armstrong, R. Virding, M. Williams, and C. Wikstrom. *Concurrent Programming in Erlang.* Prentice Hall, 1996.

[6] J. Armstrong. *Making reliable distributed systems in the presence of software errors.* PhD thesis, The Royal Institute of Technology, Stockholm, Sweden, December 2003.

[7] J. Barklund and R. Virding. *Erlang Reference Manual*, 1999. Available from http://www.erlang.org/download/erl_spec47.ps.gz.

[8] Zsók V. - Horváth Z. - Tejfel M. *Párhuzamos funkcionális programozás.* Informatika a felsöoktatásban'02. Debrecen, Aug. 28-30, 2002, pages 1085-1094.

[9] Hegedűs H. - Horváth Z. *Distributed Computing Based on Clean Dynamics.* Proceedings of the 6th International Conference on Applied Informatics, Eger, Hungary, January 27-31 2004. Vol. I. pp. 181-190.

[10] Zsók V. - Horváth Z. - Hernyák Z. *Control Language for Distributed Clean.* CSCS 2004, The Fourth Conference of PhD Students in Computer Science Szeged, Hungary, July 1-4, 2004. Acta Cybernetica, pp. 247-271.

[11] Horváth Z. - Hernyák Z. - Zsók V. *Implementing Distributed Skeletons using D-Clean and D-Box.* 17th International Workshop on Implementation and Application of Functional Languages IFL 2005 - Dublin, Ireland - September 19-21 2005