Fortran Isolates the CSE Community

Roscoe A. Bartlett, Oak Ridge National Laboratory

10/08/2013

Introduction

The continued usage of Fortran as a dominant programming language isolates the computational science & engineering (CSE) community and hinders the learning and adoption of modern software engineering methods. The issue is not primarily that Fortran is not suitable for developing many different types of software. Indeed, the are many examples of quality CSE software written in Fortran that can legitimately be considered to be "self-sustaining software" (i.e. clean design and code, well tested, minimal controlled dependencies, etc., see [1]).

The problem is that there are many CSE development groups and communities that only know Fortran and are therefore isolated from the broader software development community where Fortran usage is extremely low (ranked 24th in the TIOBE Index from Nov. 2013 [11]). Other problems with Fortran will not be discussed in the short paper (e.g. slowness to implement the 2003 standard, lack of an quality open source compiler for Fortran 2003, clumsy and inefficient text processing, etc.).

The problem is that if a Fortran programmer wants to learn about quality modern software development techniques and practices (e.g. continuous integration, unit testing, test-driven development (TDD), emergent object-oriented design, refactoring, etc. [2, 9, 8]), to what resources or references do they go? There are few (if any) quality references, training courses or instructors on these topics that are accessible to Fortran-only programmers. The issue is that all of the quality books, articles, training courses on all of these topics have examples in mainstream languages like C++ and Java and not Fortran. Mainstream languages like C++, Java, Python, etc. are so different from Fortran that they are largely inaccessible to Fortran-only CSE programmers and therefore this material is also inaccessible for Fortran-only programmers.

Example: Unit Testing

One example of the problem is the lack of unit testing and test-driven development (TDD) by Fortran-only developers. While the research and data supporting the effectiveness of unit testing and TDD has been around for many years [10, 3, 12, 7], the usage of these very successful approaches is very low in the Fortran CSE development community. This claim is backed up by personal experience, web posts, web searches, and observations by other authors. One interesting data point is that a Google search of "Fortran Unit Test" (on 10/4/2013) brings up a limited number of relevant search results but the top result is for the FRUIT Fortran Unit Test framework [5] (written in Ruby). Right on the main page for FRUIT is a quote from FRUIT's author Andrew Hang Chen which states:

> Most of the FORTRAN are important in nature, used in nuclear and aerospace codes, etc, and maintained and written actively. Please help to bring TDD practices to the FORTRAN community. The change could be very hard, personally, I quit, since I could not make the change. I hope your organization will be successful.

This is a quote from the developer and advocate of the seemingly most prevalent Fortran unit test harness software (according to Google search ranking).

Another Google search for "Fortran Unit Test tutorial" produces very little in relevant results. One exception is from NASA for pFUnit [6]. In the case of pFUnit, the tutorial shows the weakness of using Fortran for unit testing due to lack of exception handling resulting in many problems. Another example found is for fUnit [4] (also written in Ruby). The very short tutorial for fUnit describes some very rigid usage requirements and other shortcomings.

The above short survey of Fortran unit testing software and resources suggests that the sophistication and usage of unit testing software in Fortran are low compared to other more widely used modern programming languages. It is difficult to

determine why. Is it because one cannot write a quality unit test harness in raw Fortran and Fortran-only programs will not rely on a tool not written in Fortran? Is it because there is little to no demand for a quality Fortran unit test harness because those using Fortran in the CSE community have not been educated to know about unit testing and its benefits? This would be an interesting issue to research on its own.

Summary and Research Opportunities

The lack of books, articles, training courses, instructors, tutorials and tools tailored to Fortran serves to discourage the adoption of modern software development methods and is a significant factor in holding back the CSE community. The typical Fortran-only CSE developer from a non-software background lacks the foundation to go out and learn modern software development methods in the absence of such material geared toward Fortran.

However, this short paper is not arguing that all usage of Fortran is bad or is not appropriate for many different types of CSE programming projects. CSE developers who know other programming languages and have access to modern resources and tools have little problem applying good software engineering to software written in Fortran, including using test-driven development (TDD) and good unit testing. It is just that the driving tools are not Fortran based. For example, the authoer just recently wrote a bit of Fortran software that was well unit tested using TDD using a C++ unit test harness in a mixed language program. A typical Fortran-only programmer would likely not have been able to produce that level quality of testing and maintainability using existing Fortran unit test harness software.

There are two possible approaches for addressing this problem. One option is to make a significant investment in developing a set of training materials and quality tools for modern software development methods, such as for unit testing, tailored to Fortran. However, given the relatively small market of Fortran programmers and the seeming lack of interest, it is unlikely that the software community will make the investment in creating these materials and tools. The other option is to try to shift a majority of CSE projects and development groups away from Fortran as the single dominant programming language. While there are no ideal alternatives for a primary programming language for most CSE projects, one attractive approach is to simplify the usage of C++ for CSE developers. A proposal for how to do that is the focus of another position paper "Safe Domain-Specific Languages in C++11 for CSE using Compiler-Embedded Analysis Rules".

References

[1] R. A. Bartlett, Michael A. Heroux, and James M. Willenbring. Overview of the tribits lifecycle model: A lean/agile software lifecycle model for research-based computational science and engineering software. *e-science*, 2012 IEEE 8th International Conference on E-Science:1–8, 2012.

[2] K. Beck. *Extreme Programming (Second Edition)*. Addison Wesley, 2005.

[3] Thirumalesh Bhat and Nachiappan Nagappan. Evaluating the efficacy of test-driven development: industrial case studies. In *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*, ISESE '06, pages 356–363, New York, NY, USA, 2006. ACM.

[4] Jason Blevins. Fortran Unit Testing with fUnit. www2.cisl.ucar.edu/sites/default/files/pFUnitTutorial.pdf, 2012. [Online; accessed 07-October-2013].

[5] Andrew Hang Chen. FORTRAN Unit Test Framework (FRUIT). http://sourceforge.net/projects/fortranxunit/, 2012. [Online; accessed 07-October-2013].

[6] Tom Clune. Parallel Fortran unit testing framework. www2.cisl.ucar.edu/sites/default/files/pFUnitTutorial.pdf, 2012. [Online; accessed 07-October-2013].

[7] Hakan Erdogmus, Maurizio Morisio, and Marco Torchiano. On the effectiveness of the test-first approach to programming. *IEEE Transactions on Software Engineering*, 31(3):226–237, 2005.

[8] M. Feathers. *Working Effectively with Legacy Code*. Addison Wesley, 2005.

[9] R. Martin. *Agile Software Development (Principles, Patterns, and Practices)*. Prentice Hall, 2003.

[10] S. McConnell. *Code Complete: Second Edition*. Microsoft Press, 2004.

[11] TIOBE Software. TIOBE Programming Community Index for September 2013. www.tiobe.com/index.php/content/paperinfo/tpci/, 2013. [Online; accessed 07-October-2013].

[12] D. Talby, A. Keren, O. Hazzan, and Y. Dubinsky. Agile software testing in a large-scale project. *Software, IEEE*, 23(4):30–37, 2006.