**Institution: University of Kent**

**Unit of Assessment: 11 – Computer Science**

# Title of case study: Improved Functional Programming Practice through Refactoring

## 1. Summary of the impact

The programming languages and systems group at the University of Kent has built the first comprehensive tools for refactoring functional programs: HaRe for Haskell, Wrangler for Erlang. These tools not only provide a large set of refactorings, they also have facilities for managing code clones and module structure, as well as facilities for users to easily build their own refactorings.

Programmers in both open source and commercial projects use the tools to improve their programming and testing practice, and to restructure existing systems. This improves the quality of software, reducing bugs/problems for end users and cost for companies; it thus puts companies at a competitive advantage and improves best practice in industry. Evidence of take up comes from system downloads, contributions to open source repositories and company testimonials.

## 2. Underpinning research

Refactoring is the process of improving the design of a system without changing what it does. Refactoring can go hand in hand with program development - e.g. introducing a generalisation of a program component in order to reuse it - or can be performed as a part of program maintenance. For refactoring to be practicable in real-world projects, and also for it to be performed accurately, it cannot be done by hand, but needs to be supported in software by a refactoring tool.

The functional programming (FP) team at Kent, which is part of the programming languages and systems group, led by Prof. Simon Thompson (University of Kent, hereafter Kent, 1983-present), has pioneered the implementation of refactoring tools for functional programming languages, first for Haskell, with Claus Reinke (Research Fellow, Kent, 1999-2005), Huiqing Li (Research Associate, hereafter RA, Kent, 2002-05), postgraduate research students Chris Brown (Kent, 2004-08) and Nik Sultana (Kent,2007-08). More recent work has had an Erlang focus, and involved Huiqing Li (RA, Kent, 2005-now) and Pablo Lamela (RA, Kent, 2013-present), György Orosz and Melinda Tóth (Postgraduate RAs, Kent, 2007-8) and Xingdong Bian (2008-2010) and Roberto Aloi (2010-12) both Knowledge Transfer Partnership Associates with Erlang Solutions and Kent. The research has been supported by EPSRC (£311k), the EC (€1,249k) and DTI/TSB (£248k), and was done in collaboration with research groups and companies in Sweden, Spain and Hungary.

Our tools are the first practical refactoring tools for modern functional programming languages, which are increasingly used in the mainstream: for example, Erlang is currently used by Facebook, WhatsApp, Amazon, Wooga [social games maker], Last.Fm, github and Yahoo! At the core of the tools - HaRe for Haskell 98 and Wrangler for Erlang - is support for transformations of program source code, plus analysis tools that can check the complex pre-conditions on the applicability of the transformations. These transformations are not simply “cut and paste” edits, but need to respect semantics, types, module structure and so forth. Even the apparently simple refactoring of renaming an object will have complex pre-conditions for its successful application, and checking these conditions needs to be implemented too. The results are source code, and so layout and comments must be preserved if the code is to be readable by its authors. To be of practical value, the refactorings must be applicable to complete projects, and be accessible within programmers' usual working environments.

The first phase of our research was to investigate what refactoring means for functional programs, to design and implement the tools, and to make them available to the Haskell and Erlang programming communities as open source projects [1,4,5,6].

In the second phase of the research we have augmented the original systems in two ways. First we have built – at users' request – decision support tools on top of the tools to assist users in finding possible refactorings and applying them to their projects [1,3]. These tools guide users to potential problems of duplicate code or "code clones" and module structure "bad smells". The Wrangler clone detection tools have proved to be particularly useful in restructuring test code, as well as giving a basis for extracting high-level properties from sets of unit tests. In order to scale Wrangler clone detection to larger code bases, we have developed a substantially more efficient - incremental - version of the algorithm, which can be applied to an evolving project to detect new clones only in those parts of a project that have changed.

Secondly, instead of it being a "black box", we have opened up Wrangler so that it can be extended by users writing their own refactorings and conditions. This industry-leading research [1] provides the first fully open refactoring tool, which users can extend in two ways. First, we provide an API allowing users to describe new refactorings using templates written in Erlang source code; secondly, we have developed a domain-specific language to describe complex refactorings built by putting together simple refactorings. This is the first such DSL for any refactoring framework for any programming language [2].

## 3. References to the research [** - 1,3,6 best indicate the quality of the underpinning research]

[1] ** *Refactoring tools for functional languages,* S.Thompson, H. Li, Journal of Functional Programming, *23(3), pp293-350, 2013.* [Thompson REF output 4]
[2] *A Domain-Specific Language for Scripting Refactorings in Erlang* H. Li, S. Thompson, in Fundamental Approaches to Software Engineering 2012, Lecture Notes in Computer Science, 2012. [Google Scholar: 14 citations, 16/11/13]
[3] ** *Clone Detection and Removal for Erlang/OTP within a Refactoring Environment*. H. Li, S. Thompson. In Proceedings of the ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation, 2009. [Google Scholar: 33 citations, 16/11/13] [Thompson REF output 2]
[4] *Refactoring with Wrangler, updated*: H. Li, S. Thompson, G. Orosz, M. Tóth. Proceedings of the 7th ACM SIGPLAN workshop on ERLANG, 2008. [Google Scholar: 25 citations, 16/11/13]
[5] *The Haskell refactorer, HaRe, and its API.* H. Li, S. Thompson, C. Reinke, Electronic Notes in Theoretical Comp Sci 141 (4), 2005 [Google Scholar: 37 citations, 16/11/13]
[6] ** *Tool support for refactoring functional programs*. H. Li, C. Reinke, S. Thompson. In Johan Jeuring, editor, Proceedings of the ACM SIGPLAN Haskell Workshop 2003, pp 27-38. Association for Computing Machinery. [Google Scholar: 118 citations, 16/11/13]

***Funding (Thompson Principal Investigator on all grants)***
2012-2015: *PROWESS*, FP7 STREP 317820, €446k.
2011-2014: *RELEASE*, FP7 STREP 287510, €426k.
2009-2011: *Knowledge Transfer Partnership: e-learning*, EPSRC/TSB, £139k.
2008-2011: *ProTest: Property-Based Testing*, FP7 STREP 215868, €357k.
2007-2009: *Knowledge Transfer Partnership: Erlang*, EPSRC/DTI, £109k.
2005-2008: *Formally-Based Tool Support for Erlang Development*, EPSRC EP/C524969/1, £162k
2002-2005: *Refactoring Functional Programs*, EPSRC GR/R75052, £149k.

## 4. Details of the impact

Our tools, particularly Wrangler, have been used by many different individuals and organisations. Not only do these tools allow users to refactor their code in various ways, they also provide reports on different ways that the code can be improved, particularly test code, as well as providing program analyses of various kinds. Wrangler has allowed users to build better quality programs, to operate more efficiently, and to better maintain their own and legacy code, thus contributing to their profitability and sustainability, and a number have provided corroborating statements ([S1] etc.).

***Increased efficiency through defining custom refactorings in Wrangler.*** Quviq AB of Sweden use the Wrangler extension API to automate a set of custom refactorings that they use regularly. Their CTO, Thomas Arts, provides a testimonial [S1] stating that *"The investment of using the Wrangler API was well motivated. At the moment the automatic refactoring takes about 5 minutes* *to perform. It will rewrite about 1500 lines of code in 100 different files. A manual refactoring of this size is impossible to perform correctly, and would take one or two days to perform satisfactorily … we estimate a saving of up to 1 person month per year by using this Wrangler API feature."*

***Making everyday programming practice more effective.*** First, Wrangler and HaRe are available to all Erlang and Haskell programmers as Open Source tools that plug into existing widely used program development frameworks such as Eclipse, Emacs and Vim [4], and have been widely taken up (see download data below). A typical, unsolicited user comment is *"I really like Wrangler. It is one of those tools that are nice to have in the toolbox when I get to really work on some code".*

Stuart Whitfield, CEO of Erlang Solutions Ltd, who are consultants in the fields of communications, banking, messaging and transport, notes [S3] that "*it is a tool we all take for granted and enhances the nature of Erlang and how we do Erlang programming … it gives us the competitive advantage when using it as a building block in larger systems*".

Samuel Rivas, former CTO of Interoud [S2] observes "*The impact of Wrangler was to help us to produce better quality software with lower maintenance costs*. … *Wrangler was used to automate some common refactorings … these transformations are relatively easy to perform manually, but also [are] quite prone to subtle errors that are impossible when using a tool such as Wrangler*".

***Improving the quality of systems programmed in Erlang.*** Rather than interleaving refactoring and development, other users will consciously decide to tidy up a code base by making a series of refactorings. This is important, either for general reasons of code "hygiene" or to make specific preparation for code change, integration and so forth. Wrangler provides a "clone detection" facility, which reports on duplicate, or near duplicate code, in systems. Eliminating these clones leads to cleaner code, and improves the maintainability of systems.

Samuel Rivas [S2] notes: "*Wrangler helped raising the awareness of what clean code meant for us. An especially influential feature has been clone detection. After introducing it the presence of clones in code was much more evident than before, helping the team to quickly spot parts of the codebase with likely quality issues. … Wrangler also helped in making the … process to eliminate detected clones faster* ". Erlang Solutions [S3] note "*ESL staff today use Wrangler when they need to clean up code. … Once the tricky parts are working they can rely on Wrangler to efficiently refactor the code and get the general structure right.*"

***Program analysis for e-learning and code navigation.*** As well as providing support for refactoring transformations, Wrangler implements program analyses to check the pre-conditions on refactorings to ensure that they preserve program behaviour. This program analysis is also useful independently. A KTP with Erlang Solutions Ltd has developed an e-learning platform for the company to take their existing face-to-face tutorial materials online, and this was launched in October 2011. The KTP has developed a multi-modal feedback tool to give feedback not only on how a program behaves, but also its style, and this style analysis is supported by Wrangler's extension facilities. ESL [S3] note that "*this is a novel approach to e-learning … and would not have been possible without the static analysis capabilities of Wrangler*." Samuel Rivas [S2] notes that in his current position he is working with a large legacy codebase and that "*the code inspection tools [in Wrangler] can depict different levels of dependencies between functions and modules [in legacy systems] that help breaking small pieces apart … to add known APIs to increase the maintainability of the software. … Wrangler provides an easy interface for the most common scenarios that saved me and other colleagues time.*"

***Improving the quality of test code.*** Refactoring tools can be used to improve the readability of code that has been inherited from a series of coders. This is particularly acute with test code that can suffer from "copy, paste and modify" development of new tests rather than a model in which the appropriate abstractions are introduced and used. Our clone detection decision support system [3] provides the input for structure-improving refactorings dictated by test engineers with domain knowledge. The results of this approach in action with three groups of programmers and testers in Ericsson was reported at the annual Erlang User Conference, Stockholm, 2009. The payoff here is that, using the restructured code, developing new tests is substantially easier and more reliable.

Pär Emanuelsson, formerly of Ericsson, says of our work [S4] at Ericsson that "*Wrangler was used in the 4th generation Telecom system (LTE) project at Ericsson. The LTE project includes many hundreds of developers at multiple development sites at Ericsson. The system consists of many million lines of code. Wrangler was used by the testing tools groups in Linkoping and Stockholm, that is the groups that provide tools (both commercial and own developments) for the testers. These groups specialize in tools and methods for testing. Wrangler was also used by testers, who are using tools provided by the testing tools groups to test LTE products*." He concludes that Wrangler "*was valuable for improving the quality of our testing code. We have also gained valuable insights in how refactoring should be done in general on our huge applications.*"

***Property extraction.*** We have worked collaboratively with the tool developers Quviq AB to build support for the extraction of properties from existing test suites. The model here is to use clone detection to identify test cases that are similar, and to use their common generalisation as the basis for a universal property that can be tested with random data chosen from its domain of application. This functionality is incorporated in Wrangler and reported in the literature [1]. An extension of this work, under which state machine models are extracted from existing test suites is also incorporated in the tool [1]. The payoff for Quviq is that developing properties and models is made easier, thus increasing the take-up of property-based testing.

***Wider usage: download statistics and open source collaborations*** Use of Wrangler by a wider community is indicated by the download statistics, open source contributions, and github activity. On github the project has been starred by 50 users, and has 17 forks. It has 8 contributors external to the project, including a 9974 line contribution from Richard Carlsson, of Klarna AB, Sweden. Comprehensive download & contribution information is at http://www.ohloh.net/p/wrangler_refactoring/contributors/summary
.

| **Download stats** | **2008** | **2009** | **2010** | **2011** | **2012** | **2013*** |
|---|---|---|---|---|---|---|
| **Unix (linux, OS X)** | 1138 | 2611 | 2134 | 2023 | 384† | 282† |
| **Windows installer** | - | 460 | 748 | 729 | 475 | 968 |
| **Web page hits** | 7927 | 7439 | 11521 | 12388 | 13635 | 7561 |

* Up to end of July 2013. † **NB** the reduction in downloads reflects the move of Wrangler to github, https://github.com/RefactoringTools/wrangler from which we do not have download information. Most Unix users now use this.

In summary, Wrangler has had corroborated impact on the practice and effectiveness of a number of companies, as well as sustaining a broader group of open source contributors and users.

**5. Sources to corroborate the impact:** Corroborating statements have been provided by:

[S1] Thomas Arts, CTO, Quviq AB, gives a statement of the cost savings to the company coming from use of the Wrangler tool in their workflow
http://www.cs.kent.ac.uk/research/REF2014/Arts.pdf
[S2] Samuel Rivas, former CTO, Interoud, gives evidence of how Wrangler allowed the company to develop better quality software as well as supporting software maintenance activity.
http://www.cs.kent.ac.uk/research/REF2014/Rivas.pdf
[S3] Stuart Whitfield, CEO, Erlang Solutions Ltd, gives a statement on the positive effect that Wrangler had on the productivity of staff, as well as its value in large development projects.
http://www.cs.kent.ac.uk/research/REF2014/Whitfield.pdf
[S4] Pär Emanuelsson, former researcher, Ericsson, gives a statement on the value to Ericsson of Wrangler through improving the quality of testing code, as well as providing insights into the way that large software projects might be refactored
http://www.cs.kent.ac.uk/research/REF2014/Emanuelsson.txt