30 Years of Regression Testing: Past, Present and Future

Jean Hartmann
Test Architect
Office Shared Services - Engineering
Microsoft Corp.
Redmond, WA 98052
Tel: 425 705 9062
Email: jeanharm@microsoft.com

Abstract

To coincide with the 30th anniversary of PNSQC, my intent was to provide a chronology of one of software testing’s least glamorous and yet vital activities – regression testing! This short paper describes the evolution of regression testing strategies through the decades, starting out as a research topic in the early 1980’s, followed by development and early deployment by industry R&D labs in the 1990’s to adoption by software companies in the 21st century. Each decade is described in terms of the challenges faced by regression testing, projects are highlighted in which these strategies are applied and the future of these techniques is explored.

Biography

Jean Hartmann is a Principal Test Architect in Microsoft’s Office Division with previous experience as Test Architect for Internet Explorer and Developer Division. His main responsibility includes driving the concept of software quality throughout the product development lifecycle. He spent twelve years at Siemens Corporate Research as Manager for Software Quality, having earned a Ph.D. in Computer Science in 1993, while researching the topic of selective regression test strategies.
1. Introduction

Regression testing is an unsung hero of the software testing world – this indispensable testing strategy is used daily by many software companies, worldwide. Wikipedia defines regression testing as: “Software testing that seeks to uncover new software bugs or regressions in existing functional and non-functional areas of a system after changes, such as enhancements, patches or configuration changes, have been made to them. The intent of regression testing is to ensure that a change, such as a bug fix, did not introduce new faults. One of the main reasons for regression testing is to determine whether a change in one part of the software affects other parts of the software”.

The topic was first discussed in papers presented in June 1972 at the inaugural software testing conference in North Carolina, organized by Bill Hetzel [1]. References to regression testing can then be found in the classic software testing texts, such as ‘The Art of Software Testing’ [2].

At the time, the focus of regression testing was on validating mainframe software. With limited computing resources and few, if any, test automation tools available, most testing was manual. Regression test passes often took weeks to complete.

Towards the late 1970’s, researchers started to look at ways to reduce these lengthy, arduous and error-prone regression testing runs. Instead of executing all test cases every time one or more code changes were applied, known today as the retest-all approach [3], why not only run a subset of tests? The phrases regression test selection, test minimization or test prioritization were coined [4]. Thirty years later, researchers from academia and industry continue to work on the topic.

The goal of these regression test selection strategies remains the same:

- Determine a subset of tests for less cost than rerunning all tests [3]
- Ensure that this subset is ‘safe’ aka can catch product bugs compared to retest-all

The remainder of this paper chronicles the story of regression testing and in particular, regression test selection since 1980. It highlights key advances made by both academia and industry in developing efficient and effective regression test selection techniques. It demonstrates the time taken for techniques to transition from theory into practice. Sections 2 through 4 discuss how the technique has evolved through the decades. In Section 5, we examine how it is influencing testing today and discussing its future role.
2. 1980s: Genesis

In the early 1980’s, Fischer [5,6] was the first to explore the topic of regression test selection. He applied mathematics in the form of operations research to model the problem, which is known as a set-covering problem [7]. The application of linear or integer programming techniques allowed him to identify a subset of test cases that provided coverage of modified program code, ensuring that each modified code entity was exercised by at least one test case. The model he used consisted of:

- **Objective function** that represented each test case and its associated cost
- **Set of constraints** that reflected each test case that executed a code entity

The set-covering problem is known to be NP-hard, meaning that in theory, it cannot be solved in polynomial (aka a reasonable amount of) time. In practice, therefore, there was no guarantee that the model solver (tool) Fischer used could determine an optimal solution to any given problem. Computing resources at the time were limited and so applying his strategy to large-scale problems was not feasible.

This minimal solution to selecting a subset of tests was also not deemed ‘safe’ by the testing research community at large. They preferred to explore other approaches that picked more tests for rerun in order to raise the likelihood of finding the same number of bugs compared to the retest-all approach. With regards to picking a safe or more conservative set of tests, Fischer’s work represented the riskiest of the different approaches. Nevertheless, his work represented a milestone in regression testing research and would be vindicated in later years.

In the mid to late 1980s, when software engineering, and especially software testing, established itself as an important part of the software development lifecycle, there was renewed interest in regression testing. Researchers, who had developed new testing strategies, such as data-flow based testing [8], combined them with impact or ripple effect analysis techniques to select tests based on those changes [9,10].

However, these strategies had serious limitations in practice. They could not easily scale to the testing of large code bases. Dataflow-based techniques, which were used by many researchers, required the computation and analysis of so-called def-use pairs prior to test execution [11]. Each pair represented the definition and use of a program variable. Computing such pairs for large, real-world code bases required the use of complex code analysis techniques that were expensive in terms of compute time. These techniques would become even more complex when analyzing code containing numerous code functions [12] and handling language constructs, such as pointer variables, which were being introduced with the C programming language at the time [13]. These analysis techniques were known as intra- and inter-procedural analysis. In addition, the majority of the dataflow-based regression test selection techniques did not necessarily derive a minimal subset of tests.

Meanwhile, in industry, the majority of applications were now being developed in C. Code was increasing in size and complexity. The introduction of the first test automation
tools enabled companies to move from manual to automated testing scenarios to improve test reliability. Automated regression test suites began to grow in size, raising the need for efficient and effective regression test selection strategies.

In summary, the 1980’s saw the genesis of regression testing research. While researchers could develop theoretical approaches, they could only explore their practical application by means of toy examples. Any tools too that were developed to support the proposed techniques were very limited in their analysis capabilities, especially when faced with the newer and more complex programming language constructs being introduced.

In the next decade, these techniques become more mature – existing approaches were refined, more industrial-strength tools and test suites were built to validate techniques. The 1990s became the decade of empirical studies for regression testing.

3. 1990s: Putting Theory into Practice

The 1990’s saw regression test selection theory being put into practice. Researchers, realizing the high cost of developing tools to adequately support their investigations, turned to industry for support. They started collaborations with the large industrial research laboratories such as Bell Labs [14] Bellcore [15] and Siemens [16] to develop tools to automate the collection and analysis of the required data. Some academic research centers pursued similar initiatives, but at a lower cost and with fewer capabilities. They typically built tools on top of public domain compilers such as the GNU compiler resulting in toolsets, such as Combat [17].

Using these various tools, the teams launched extensive empirical studies in which larger pieces of code, typically C code from the public domain, were targeted for analysis and test suites were developed to exercise and revalidate them [18]. Such studies typically seeded faults in the program code, deemed them adverse program modifications, and then applied the respective revalidation strategy to demonstrate the ability to select a subset of tests and safety of that subset with respect to fault detection.

A major factor enabling the development of such tools was the availability of powerful, yet affordable, personal workstations such as those developed by Sun Microsystems. Running the required code analyses, which a decade earlier had practical limitations in terms of scalability and performance on early PCs, were now possible. Both academia and industry also leveraged this computing power to address the complexities of analyzing a new programming language - C++. Object-orientation, with concepts such as inheritance and templates, introduced additional challenges for regression testing researchers. They needed to adapt their established techniques and tools [19].

During the early 1990’s, researchers began to revisit Fischer’s approach, applying algorithms that could now overcome the theoretical limitations of NP-hard problems [20,21]. These so-called greedy algorithms represented heuristics that could provide a
solution quicker than model solvers, but their solutions were not necessarily minimal or optimal.

By the mid-1990s, Siemens researchers, encouraged by the results of these empirical studies and advances in tools, encouraged their industrial business units to explore the potential benefits of regression test selection. One Siemens division that researchers believed stood to benefit from this regression testing technology was the telecommunications division, which developed large public telephone switches (including the required software) for the US and world markets.

Their system test passes, by now largely automated, took approximately six weeks to complete. However, during that time, code fixes or updates/patches were made to the switch software on a regular basis, but the test pass could not be stopped and restarted, because of time and cost constraints. Therefore, the team needed a strategy and tools that could quickly identify the impact of code changes and determine a minimal subset of system tests for those changes. Siemens researchers responded by building a custom toolset based on Fischer’s work that identified a subset of tests for rerun and kept the test pass within schedule. It was an early example of where regression test selection was to be used over the next decade – optimizing post-checkin validation or system testing processes where large, automated test runs needed to be streamlined.

One distinct twist in this regression testing evolution was its application to hardware or IC testing. Test suites in the hardware world, or so-called testbenches, typically contain large numbers of test cases or test vectors, for validating large ASICs or custom ICs. Companies, such as Intel, are always looking for ways to make chip testing more efficient. In the late 1990’s, a small, U.K. startup company called TransEDA, which was developing code coverage analysis tools for Verilog and VHDL, incorporated regression test selection strategies into their VN-Optimize tool [22]. Based on this technology, hardware companies were able to prioritize or reduce the number of test cases or vectors they ran when modifying their Verilog or VHDL code.

In summary, the 1990’s were a decade of increased practicality where researchers demonstrated that their regression test selection approaches were applicable to ‘real’ code. For the big industrial research labs, it was a time of building the tools that could automate the most promising of these research techniques and exploring the transfer of this technology to its business units to provide a competitive advantage.

4. 2000s: Large Scale Application

By the start of the new millennium, the focus of most industrial research labs was shifting to more development work and budgets were being curtailed. Developing leading edge testing technologies, such as those associated with regression test selection, were hard to fund. Only one major player with sufficient resources remained to continue the efforts – Microsoft.
The company had developed an in-house toolset known as Magellan with the help of a research group within Microsoft Research [23]. This industrial-strength toolset included a tool known as Echelon (now called Scout), which codified a greedy algorithm for selecting tests, vindicating the minimization approach. The effort was being driven by the need that the number of automated test cases and thus regression testing costs were exploding. Running and analyzing the results from frequent test passes, whether automated or manual, was very time-consuming and costly.

The Windows Sustained Engineering (SE) team was one of the first teams to apply this in-house regression test selection toolset. The team was responsible for validating maintenance changes to large operating system code bases after they had shipped. Today, these include Vista and Windows 7, together with corresponding Windows server operating system releases. Test suites for these operating system releases run into the hundreds of thousands of tests, both automated and manual. Correspondingly, test passes can take weeks to complete.

By applying the Magellan Scout tool, this team was able to significantly reduce the number of regression tests being rerun, making their maintenance update cycles much faster and more efficient. The team also found that selecting minimal sets of tests did not necessarily affect safety in terms of detecting fewer faults than running all tests again.

Over the years, other teams within Microsoft have also explored the technology with success [24]. The Scout tool keeps evolving to the point where many of the limitations that were discussed at the start of this paper, are no longer valid. A good example of this is the way in which tests are selected. In the latest version of the Scout tool, the original heuristic has been replaced by the latest generation of Microsoft model solvers, which are now capable of solving large and complex linear programming problems in reasonable timeframes and with optimal results. This enables ever larger Microsoft test suites to be processed.

5. 2010 and Beyond

Today, regression test selection lives on and its prospects look promising. With the growth of online services and the need for moving software quality upstream in the development process, regression testing is being explored in the context of pre-checkin validation. Developers are creating large numbers of unit test cases, which they cannot all run, before checking in their code. Check-ins are frequent and need to be fast to enable developers to quickly implement and test their features. Regression test selection tools, such as Scout, could provide developers with a viable option of validating only their changed code and reducing the number of unit tests they need to rerun.

The concept of regression test selection has also found its way into Microsoft products. For example, Visual Studio 2010 and 2012 contain a feature known as the Test Impact
Analyzer [25]. Unlike the Scout tool, it does not select a minimal set of tests to rerun, but instead selects an aggregate set, which includes all tests traversing the modified code.

6. Conclusion

This paper has provided a very brief insight into a thirty year history of regression testing, highlighting major research directions, milestones, tools and researchers. It chronicles the evolution of regression testing from research to large-scale industrial application.

7. Acknowledgement

I would like to thank Curtis Anderson, Principal Test Manager of Office Engineering Test, Tara Roth, Corporate VP of Office Shared Services for their ongoing support. My sincere gratitude also goes to Kurt Fischer, Executive Vice President EM&I, my Ph.D. supervisor David Robson and many of my fellow software testing researchers, colleagues and friends who inspired me to explore the topic, guided and challenged me. Special thanks to Prof. Lee White, Prof. Mary-Jean Harrold, Dr. Tom Ostrand and Dr. Elaine Weyuker.

8. References

17. M.J. Harrold and P. Kolte, "", PNSQC, 1992
22. VN-Optimize (http://www.sereda-tech.co.uk/transeda-vn-optimize.php)