Software Engineering Solutions for Improving the Regression Testing Methods in Scientific Applications Development

BOJANA KOTESKA, LJUPCO PEJOV AND ANASTAS MISHEV, University SS. Cyril and Methodius

The aim of this paper is to improve the testing process of scientific applications by proposing some software engineering solutions for regression testing. Given the fact that changes are a common part of the development of scientific applications the need for regression testing is essential. Concerning to improve the methods of regression testing the relationship between requirements and test case selection for regression testing is included as a relevant for improving the overall quality of the testing process. Special emphasis is given on the requirements' changes during the application development process and their impact on test case modifications. In addition, we also state the reasons for the implementation and the importance of the regression testing as a part of the verification process of scientific applications. In order to get more relevant results we made an interview with a scientist (chemist) about the regression testing practices in scientific applications. The conclusions and recommendations in this paper are based on the analysis of the results of a survey conducted among scientists in the HP-SEE project and the answers of the interview questions.

Categories and Subject Descriptors: D.2.4 [**Software Engineering**]: Software/Program Verification —*Validation, Formal Methods*; G.4 [**Mathematics of Computing**]: Mathematical Software—*Certification and testing, Documentation, Verification*

General Terms: Verification

Additional Key Words and Phrases: Regression testing, scientific application, software engineering, software quality

1. INTRODUCTION

Advances in research in scientific areas continually impose the need for creating scientific applications. One of the definitions of scientific applications describes the scientific application as a mathematical model that simulate the natural phenomena [PcMag 2013]. The need for creating large and complex mathematical models mathematical calculations increases the probability of errors and thereby increases the need for performing changes in different parts of the application. Successful testing after a change ensures that the system is in a stable condition and it provides assurance that the system did not lose any functionality, but only new functionalities are added or the incorrect ones are replaced. Taking into account the fact that changes are a common part of a scientific application development process because of the constant changes in requirements and source code the need for regression test-

This work was supported in part by the European Commission under EU FP7 project HP-SEE (under contract number 261499).
Author's address: Bojana Koteska, University SS. Cyril and Methodius, Faculty of Computer Science and Engineering, Rugjer Boshkovikj 16, P.O. Box 393 1000, Skopje; email: bojana.koteska@finki.ukim.mk; Ljupco Pejkov, University SS. Cyril and Methodius, Faculty of Natural Science and Mathematics, P.O. Box 1001, 1000 Skopje; email: ljupcop@iunona.pmf.ukim.edu.mk; Anastas Mishev, University SS. Cyril and Methodius, Faculty of Computer Science and Engineering, Rugjer Boshkovikj 16, P.O. Box 393 1000, Skopje;email: anastas.mishev@finki.ukim.mk.

In: Z. Budimac (ed.): Proceedings of the 2nd Workshop of Software Quality Analysis, Monitoring, Improvement, and Applications (SQAMIA), Novi Sad, Serbia, 15.-17.9.2013, published at http://ceur-ws.org

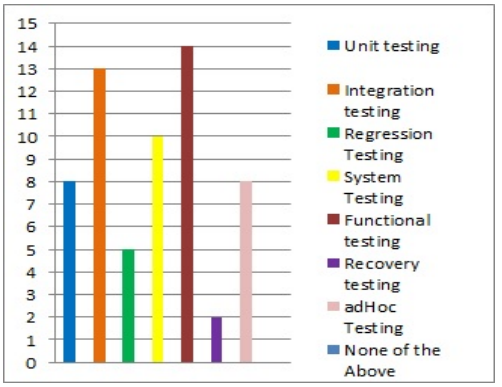

Fig. 1. Answers to the question: "Which types of testing do you perform?".

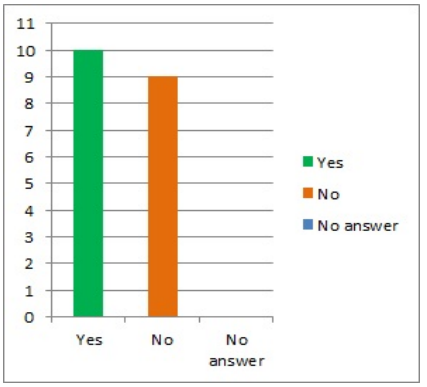

Fig. 2. Answers to the question: "Do you link requirements to tests that verify them?".

ing is essential [Letondal and Zdun 2003]. For example, the results obtained from physical experiments can be the reason for making changes in the requirements and application source code.

The main motivation for this research comes from the results obtained from the survey we conducted among 20 scientists in the HP-SEE [Koteska and Mishev 2013] (High-Performance Computing Infrastructure for South East Europe's Research Communities) project. The scientific applications in this project are organized in three scientific research communities: Computational Physics, Computational Chemistry and Life Sciences. There are total 23 applications [HP-SEE 2013].

The purpose of this paper is to improve the testing process of scientific applications, especially the regression testing. Some of the most common software engineering practices are pointed out and several recommendations about each one are given. The main contribution is that we adapt some existing software engineering practices which are used for commercial software development where real customers exist by making small modifications. The difference between scientific and commercial software testing is in the testing methods. For example, one of the key differences is the definition of requirements and inability the both kind of software to be tested in the same way. Given that in the scientific software development, real customers do not exist, checking the accuracy of scientific applications is very difficult. Usually, to check whether the software satisfy a requirement, it is necessary to perform a real experiment [Segal].

The most important topics are the regression testing and the relationship between requirements and tests. In order to show the current practice of scientists in the project the next two questions were chosen as the most relevant to these issues. The results to the question about testing types which scientists performed in the application development process are shown on Fig.1. They show that most of the scientists performed integration and functional testing, but only five development teams performed regression testing and two teams perform recovery testing. Fig. 2 presents the results of the question about linking requirements and tests. The majority of the scientists did not define requirements or define them in inappropriate form therefore the fact that almost half of the development teams used to link requirements to tests is questionable.

2. BACKGROUND

Regression testing is defined as a repeated execution of test suites in order to ensure that the system does not fail after a modification and that test suites still pass [Christensen 2011]. In [Singh et al. 2010], the authors describe some regression testing techniques. They mentioned three categories of re-

gression testing techniques: selection, test prioritization and hybrid techniques. Selection techniques choose some tests from the old test suite and execute them on the new modified version of the software application. Test prioritization reorders test suite in order to improve the effectiveness of testing. Hybrid techniques combine both the selection and test prioritization techniques. An approach for prioritization of test cases that should be used in regression testing process is presented in [Srikanth et al. 2005]. Test case prioritization is based on four factors: requirements volatility, customer priority, implementation complexity, and the fault proneness of the requirements. This approach is not fully applicable to scientific software development because real customers do not exist. The factor ”customer priority” must be removed. A test suite prioritization algorithm is given in [Srivastava et al. 2008]. This algorithm prioritizes the test cases with a goal of maximizing the faults that can occur during the source code execution. In [Remmel et al. 2012], the authors proposed a regression test environment, an infrastructure for testing, at system testing level for DUNE (the Distributed and Unified Numerics Environment). DUNE is a free software framework for solving partial differential equations with grid based methods [Bastian et al. 2008]. They create tests which support algorithm verification and scientific validation. In [Kelly and Sanders 2008], the authors mention the regression testing as a factor for improving the quality of scientific software development. An advice for turning the bugs into test cases is given in [Aruliah et al. 2012]. This approach can be used for building regression tests. The bad regression testing practices are presented in [Sanders 2008]. According to this paper, scientists usually do not perform systematic regression testing, but they use ad-hoc regression testing. Also, they use the same input data for regression tests for more than a year. The regression tests that are consisted of comparison of simulation results with reference results are not practical according to [Remmel et al. 2011]. They are used for ensuring the experimental results, but they should be avoided because the differences in the outputs may be caused by an enhancement in the code. According to the survey results presented in [Heaton et al. 2012]], regression testing is important for scientists, but they do not perform it correctly because they are not familiar with it. Also, the developers do not use automated regression testing. The other thing important for this research is the connection between the requirement changes and test case selection for regression testing. An approach to regression test selection which uses system requirements and their associated test cases instead of source code is presented in [Chittimalli and Harrold 2008].

3. THE INTERVIEW

In order to get more relevant results, we did an interview with a scientist (chemist) who has big experience in developing scientific applications and one of the participants in the HP-SEE project. There are total 12 questions that mostly refer to requirements changes, test cases and regression testing. The short version of the interview is presented in this section.

Q1: How often do changes in requirements occur during the application development?

A1: There are two possible reasons for changes in requirements in the course of application development. The first one is related to the new scientific issues that might have arisen during the research, the second one, on the other hand, comes into play when the researcher comes to (or invents) a new approach which is superior to the one that has been used up to that point.

D1: Advances in research and new results obtained from experiments are the most common reasons that lead to changes in scientific applications. Scientists know that the possibility of such changes is great. Consequently, it is necessary to make changes in several segments of the code, while not breaking parts that perform correct calculations. Also, it is necessary to keep the concept of the functioning of the system as a whole.

Q2: How do changes in requirements affect the source code?

A2: If only a minor change in the algorithm or addition of another variable that needs to be computed is required, often small changes in the source code are required. On the contrary, when a new algorithm or approach is intended to be included in research, often more profound changes to the source code are required, and it certain cases it is much easier to write the code from scratch than to introduce changes in the current version.

D2: Problems faced by scientists when they need to make major changes in the algorithm commonly occur because of an inadequate documentation and poor visibility of the source code. Sometimes they do not understand the code that had developed before a certain period of time. Therefore, it is easier for them to develop a new algorithm for solving the same problem.

Q3: When do most of the changes occur (at which stages of development)?

A3: After the code has been tested on a wide variety of systems. In this phase, often most of the ”bugs” are identified, or some errors (both conceptual and trivial) are seen. However, also numerous changes can occur at any point at which a change in requirement occurs.

D3: Usually scientists perform testing after a certain period of time when there are many written lines of code. This approach causes errors to be discovered too late, and thus increases the number of changes that need to be made. Small changes which are made as a result of a change in any one of requirements that are not specified in the appropriate form, are not recorded usually.

Q4: How much time do you need to make a source code correction after each change made?

A4: It depends on the complexity of the correction, addition, or functionality enhancement. In most cases, especially if only minor changes are required, source code modifications are quick. In cases, however, where new conceptual changes are introduced, or new functionality in the code is included, the source code changing may take an appreciable amount of time.

D4: Changes that occur as a result of the addition of new functionalities in the code or change in the conceptual model are usually time-consuming. In particular, this happens if the code is infinite, badly written and undocumented.

Q5: Do you perform regression testing ? If yes, how do you perform it?

A5: I often do, unless the change is exceptionally small or pretty trivial. I usually repeat certain calculations on some sample systems or on cases which I have treated before, or some prototype systems (e.g. analytically-solvable systems).

D5: Scientists usually preform testing after changes, but not always on time. For example, several changes can be made in the source code and after that when testing is performed an error can be found. In this case, it can be difficult to determine the reason for its occurrence.

Q6: Do you avoid regression testing after the change and if you avoid it, why?

A6: I usually do not insist on avoiding regression testing (unless the change is really trivial, e.g. improved value of a fundamental or other constant).

D6: This testing method is usually used by scientists, but the changes are not documented and a previous version of the application is not kept. Sometimes testing after a big change can give accurate results, but later it may be necessary to return to the previous stable version of the application.

Q7: Do you make test case change after changing application test requirements associated with those test cases? If not, why?

A7: Changes in test cases are required if there are substantial changes in the functionality of the code, e.g. if the code applicability is extended to a more flexible group of systems, or if certain new concepts are introduced.

D7: Scientists usually do not create test cases for each requirement because they do not write a proper requirements specification. If a change in a requirement occurs and it is not propagated in the corresponding test cases, then the possible errors may not be noticed.

Q8: How do you make test cases?

A8: The test cases are usually either simple, prototypical systems, for which the required parameters are either known or can be computed by e.g. analytical approach.

D8: The answer shows that the tests are usually performed with the help of systems that already have known analytical solutions. This approach makes the process of specification of test cases difficult.

Q9: Does inadequate testing adversely affect the application development process?

A9: Sure, as this can affect both the code concepts and functionality.

D9: Scientists are aware that the wrong way of testing and insufficient testing can disrupt the functionality of the application. Improving the way of testing applications can greatly increase the quality of the applications.

Q10: Do you use any tools in the testing process and if you do, what type of tool it was?

A10: No.

D10: There are tools which create test cases, run tests, keep records of test cases and automate much of the work that is carried out manually. The lack of using testing tools during the development process can increase the time required for testing and time required for developing the application.

Q11: Do you record all the changes made to the requirements and the source code? Do you have any documentation about that? Do you use some templates?

A11: I only make comments throughout the source code. I usually make no documentation for my highly specialized codes. However, if one intends to make a more flexible code, for a more general purpose, then the actual usage of the code will be highly dependent on the quality of the documentation.

D11: Certainly, the main goal of the development process is not the creation of documents, but they serve as an additional support mechanism for dealing with changes in requirements and source code. There are a number of templates offered by software engineering which with minor modifications will fit perfectly in the development of scientific applications.

Q12: Do you think that regression testing is important?

A12: It certainly is. It can help in both the time line of the application development, and also in its correctness.

D12: The regression testing improves the quality of applications by ensuring that adding a new functionality or changing the existing will not impair the functionality and correctness of the application.

4. SOFTWARE ENGINEERING SOLUTIONS FOR REGRESSION TESTING

4.1 Definition of Requirements

Before the process of development starts, scientists need to define the application requirements. All requirements cannot be predicted in advance because it depends on the progress in research. Also, changes in requirements occur very often. A good practice is to define the requirements in iterations. The iterative development is recommended for the entire application development process because the requirements need not be completely understood and specified at the start of the development process and they can be added in any iteration [Jalote et al. 2004]. It is necessary to specify the expected results for each functionality. If it is not possible to predict the exact value, then it is necessary to define the range of values where the solution lies. A domain specific requirements model for requirements is presented in [Li et al. 2011]. The model includes the scientific knowledge as a key factor for identifying requirements and it provides domain specific requirement model elements and associations between them. should be specified using templates proposed by software engineering. Also, some modifications to the template should be made if necessary.

Each requirement should contain the following basic fields: Id - unique requirement identifier; Name - short name of the application that reflects the functionality that needs to be realized; Type of appli-

cation - functional, non-functional or domain request; Version - current version of the requirement; Description - a detailed description of the functionality that needs to be realized (Description of the resources required to execute the application); History of the changes in each version of the requirement.

The process of defining requirements will help in the test cases specifications and in the process of testing the application. Well specified requirements lead to a more precise definition of the test cases and better organization of the overall development process.

4.2 Definition of Test Cases

Test cases can be generated automatically by using a static approach which generates inputs from some kind of model of the system (model-based testing) and a dynamic approach which generates test cases by executing the program repeatedly [Artho et al. 2005]. The main thing that needs to be considered while generating test cases is to create at least two test cases for each requirement. Moreover, the first described test case should return the correct value (the one that is expected), and the second described test case should return an incorrect value. Test cases where a correct value cannot be specified should contain a range of values.

Each of the descriptions of the test cases should include the following fields: Id - a unique test case identifier; Version - the current version of the test case; Name - name of the test case; Requirement id - id of the requirement that is connected to this test case; Goal -a description of the goals of the test case; Conditions - conditions that must be met in order to perform the test case (results from other tests or some additional conditions); Execution environment of the test case; Expected results; History of the changes in each version.

Research has shown that scientists do not use any additional tools that automate the process of testing and comparing the outputs with the expected results. This is particularly important when regression testing is performed because the same test cases are being repeated.

4.3 Test Cases Selection

The selection of test cases before the process of regression testing starts is one of the most important prerequisites for quality testing. The decision about test cases that should be selected and re-executed after a change depends on many factors. The selection of test cases that need to be performed again included those test cases that are related to the performed change. A good test cases selection technique eliminates the redundant test cases and assigns priority to test cases such that test cases with higher fault-detection capability are assigned a higher priority. Some of the most common selection techniques for procedural programs are: dataflow analysis-based techniques, slicing-based techniques, firewall-based techniques, differencing-based approaches, control flow analysis-based techniques [Biswas et al. 9 01].

The process of the execution of the relevant test cases can be split into three categories: test cases that relate to the requirement that had been changed; test cases associated with the module from the source code which had been changed; Execution of the test case that had been changed. The second option is the most difficult to be performed- to find the test cases when a change was made to the source code. Then the functionality of that part of the code needs to be identified and the appropriate test cases associated with it need to be found. For easier identification of test cases that relate to a particular change in the code, it is necessary to organize the code into smaller independent units - functions. A good selection technique of test cases requires proper specification of the application requirements and test cases, and maximum visibility and organization of the source code parts. Also, if a test case depends on other test cases and the change was performed in that test case it is necessary to execute all test cases that it depends on.

4.4 Regression Testing

Some of the main reasons for regression testing are: adding or changing a functionality and bug detection, fixing defects, adding or adapting the existing functionalities or porting the software to different environments [Biswas et al. 2010]. Another reason for regression testing is refactoring. In [Rachatasumrit and Kim 2012], authors showed the relationship between refactoring edits and regression testing by applying a change impact analysis and a refactoring reconstruction analysis.

A good approach is to use the hybrid technique defined in [Wong et al. 1997] which combines the modification, minimization and prioritization-based selection using a list of source code changes and the execution traces from test cases run on previous versions.

The process of regression testing requires several additional things that must be done before the process of testing starts. The activity diagram of the steps before and after the regression testing process when a new functionality should be added or an existing should be changed is shown on Fig. 3. After changing a requirement, when a new functionality is required or the existing one should be changed, first thing that needs to be done is test case correction. It means that all test cases affected by the requirement change must be modified. Also, the appropriate modifications should be done in the source code.

One of the most important thing is the selection of a set of test cases for regression testing. In this case, the changed test cases must be included in the set. After this step, the regression testing should be performed. If all test cases pass then the regression testing is performed successfully. If not, then all test cases changes should be reviewed and the next steps should be repeated.

The Fig. 4 presents the steps before and after regression testing when a bug in the application is found. After the bug is found, the reasons for that must be clearly indicated and adequate changes should be made in the source code. Before the testing is performed the selection of the test cases should be done. It is not good practice to take into consideration all test cases during the regression testing because not all test cases are affected by the found bug. If any of the tests fail then the reasons for the bug should be considered again and all steps after should be repeated.

The regression testing is a very important step of the development process of scientific applications. The realization of the regression testing mostly depends on the specification of requirements and test cases. Since when developing scientific applications is not possible to define all requirements at the beginning a good practice is the process of testing to take place in stages (iterations). Moreover in each iteration the requirements that have been implemented or changed should be tested. This type of testing helps in reducing the number of errors that occurred as a result of a change of requirement or a change in the source code of a scientific application. The main reason for the implementation of the regression testing is that it can improve the quality of the whole testing process and the application.

5. CONCLUSION AND FUTURE WORK

In this paper we have presented some useful software engineering solutions for regression testing. The results obtained from the survey we conducted among scientists in the HP-SEE project and the interview showed that the regression testing is a fundamental part of the scientific application development process. We gave an overview of the most common reasons for changes that occur during the development of the scientific applications. Also, the most important steps of the regression testing were presented. This paper outlines the software engineering practices that should be included in the development process in order to improve the quality of the testing process and overall scientific application.

Our future work is oriented to developing a framework for scientific application development, where more specific software engineering practices will be given for each development phase and our future

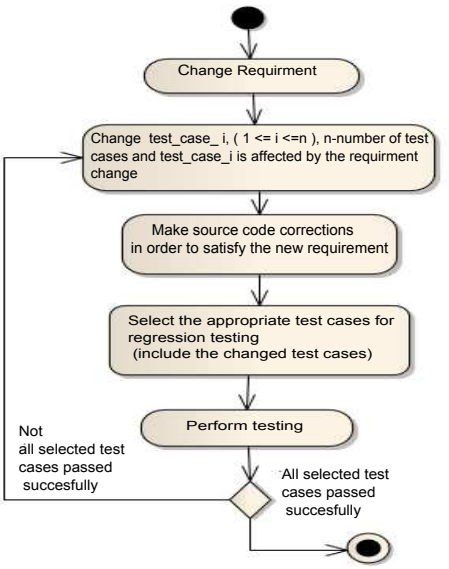

Fig. 3. An activity diagram of regression testing process when a new functionality is added or the existing is changed

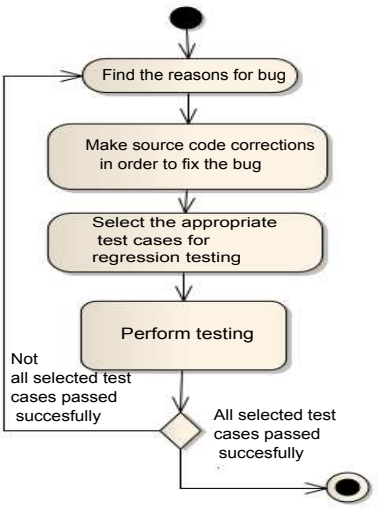

Fig. 4. An activity diagram of regression testing process when a bug is found

research is aimed at proving this concept in practice. It means that all these recommendations and practices will be used in the development process for a specific scientific application. The purpose of this framework is to provide a detailed guide for scientists when developing scientific software. It will give a detailed software engineering solutions for each development phase (defining requirements, programming, testing, etc.) and also some development techniques. It will help scientists to learn some common software engineering practices. The framework will include the software engineering practices for scientific software development, but also in order to adapt to the scientific software development process, some modifications of the existing software engineering practices for the commercial software development will be made.

REFERENCES

Cyrille Artho, Howard Barringer, Allen Goldberg, Klaus Havelund, Sarfraz Khurshid, Mike Lowry, Corina Pasareanu, Grigore Rosu, Koushik Sen, Willem Visser, and Rich Washington. 2005. Combining test case generation and runtime verification. *Theor. Comput. Sci.* 336, 2-3 (May 2005), 209–234. DOI:http://dx.doi.org/10.1016/j.tcs.2004.11.007

D. A. Aruliah, C. Titus Brown, Neil P. Chue Hong, Matt Davis, Richard T. Guy, Steven H. D. Haddock, Katy Huff, Ian Mitchell, Mark Plumbley, Ben Waugh, Ethan P. White, Greg Wilson, and Paul Wilson. 2012. Best Practices for Scientific Computing. *CoRR* abs/1210.0530 (2012).

P. Bastian, M. Blatt, A. Dedner, C. Engwer, R. Klofkorn, M. Ohlberger, and O. Sander. 2008. A generic grid interface for parallel and adaptive scientific computing. Part I: abstract framework. *Computing* 82, 2 (July 2008), 103–119. DOI:http://dx.doi.org/10.1007/s00607-008-0003-x

Swarnendu Biswas, Rajib Mall, Manoranjan Satpathy, and Srihari Sukumaran. 2010. Regression Test Selection Techniques: A Survey. (2010).

Swarnendu Biswas, Rajib Mall, Manoranjan Satpathy, and Srihari Sukumaran. 2011-09-01. Regression test selection techniques: a survey.(Report). *Informatica* 35, 3 (2011-09-01), 289(33).

Pavan Kumar Chittimalli and Mary Jean Harrold. 2008. Regression test selection on system requirements. In *Proceedings of the 1st India software engineering conference (ISEC '08)*. ACM, New York, NY, USA, 87–96. DOI:http://dx.doi.org/10.1145/1342211.1342229

H.B. Christensen. 2011. *Flexible, Reliable Software: Using Patterns and Agile Development*. Taylor & Francis. http://books.google.mk/books?id=VQaf_vOTDzMC

Dustin Heaton, Jeffrey C. Carver, Roscoe Bartlett, Kim Oakes, and Lorin Hochstein. 2012. *The Relationship between Development Problems and Use of Software Engineering Practices in Computational Science & Engineering: A Survey*. Technical Report SERG-2012-05. Department of Computer Science, The University of Alabama. http://software.eng.ua.edu/reports/SERG-2012-05

HP-SEE. 2013. HP-SEE officail website. http://www.hp-see.eu/. (2013). Accessed March 22, 2013.

Pankaj Jalote, Aveejeet Palit, and Priya Kurien. 2004. Timeboxing: A process model for iterative software development. *Journal of Systems and Software* 2004 (2004), 3.

Diane Kelly and Rebecca Sanders. 2008. Assessing the Quality of Scientific Software.

Bojana Koteska and Anastas Mishev. 2013. Software Engineering Practices and Principles to Increase Quality of Scientific Applications. In *ICT Innovations 2012 (Advances in Intelligent Systems and Computing)*, Smile Markovski and Marjan Gusev (Eds.), Vol. 207. Springer Berlin Heidelberg, 245–254. DOI:http://dx.doi.org/10.1007/978-3-642-37169-1_24

Catherine Letondal and Uwe Zdun. 2003. Anticipating Scientific Software Evolution as a Combined Technological and Design Approach. In *in Second International Workshop on Unanticipated Software Evolution (USE2003*. 1–15.

Yang Li, N. Narayan, J. Helming, and M. Koegel. 2011. A domain specific requirements model for scientific computing: NIER track. In *Software Engineering (ICSE), 2011 33rd International Conference on*. 848–851. DOI:http://dx.doi.org/10.1145/1985793.1985922

PcMag. 2013. Definition of scientific application. http://www.pcmag.com/encyclopedia/term/50872/scientific-application. (2013). Accessed March 20, 2013.

N. Rachatasumrit and Miryung Kim. 2012. An empirical investigation into the impact of refactoring on regression testing. In *Software Maintenance (ICSM), 2012 28th IEEE International Conference on*. 357–366. DOI:http://dx.doi.org/10.1109/ICSM.2012.6405293

Hanna Remmel, Barbara Paech, Peter Bastian, and Christian Engwer. 2012. System Testing a Scientific Framework Using a Regression-Test Environment. *Computing in Science and Engg.* 14, 2 (March 2012), 38–45. DOI:http://dx.doi.org/10.1109/MCSE.2011.115

Hanna Remmel, Barbara Paech, Christian Engwer, and Peter Bastian. 2011. Supporting the testing of scientific frameworks with software product line engineering: a proposed approach. In *Proceedings of the 4th International Workshop on Software Engineering for Computational Science and Engineering (SECSE '11)*. ACM, New York, NY, USA, 10–18. DOI:http://dx.doi.org/10.1145/1985782.1985785

R. Sanders. 2008. *The Development and Use of Scientific Software*. Queen's University (Canada). http://books.google.mk/books?id=9L7jKBqZpm0C

Judith Segal. Scientists and software engineers: A tale of two cultures. http://oro.open.ac.uk/17671/

Yogesh Singh, Arvinder Kaur, and Bharti Suri. 2010. A Hybrid Approach for Regression Testing in Interprocedural Program. *JIPS* 6, 1 (2010), 21–32.

H. Srikanth, L. Williams, and J. Osborne. 2005. System test case prioritization of new and regression test cases. In *Empirical Software Engineering, 2005. 2005 International Symposium on*. 64–73. DOI:http://dx.doi.org/10.1109/ISESE.2005.1541815

Praveen Ranjan Srivastava, Krishan Kumar, and G. Raghurama. 2008. Test case prioritization based on requirements and risk factors. *ACM SIGSOFT Software Engineering Notes* 33, 4 (2008).

W.E. Wong, J.R. Horgan, S. London, and H. Agrawal. 1997. A study of effective regression testing in practice. In *Software Reliability Engineering, 1997. Proceedings., The Eighth International Symposium on*. 264–274. DOI:http://dx.doi.org/10.1109/ISSRE.1997.630875