Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 1 (2012) 1505–1509
Procedia Computer Science
www.elsevier.com/locate/procedia

International Conference on Computational Science, ICCS 2010

(Position Paper)Applying Software Engineering Methods and Tools to CSE Research Projects

Hoda Naguib, Yang Li[1,*]

Institut für Informatik, Technische Universität München Boltzmannstr. 3, 85748 Garching, Germany

Abstract

The need for applications that are developed especially for Computational Science and Engineering (CSE) has been growing rapidly in the recent years. These applications are often a prerequisite for research and have to be evolved and maintained for considerable periods of time. However, CSE researchers have traditionally put focus on achieving better computational performance and results rather than the software's comprehensibility, maintainability and extensibility. This paper first presents two case studies on two different CSE research projects, where common and specific problems are identified. Second we propose solutions that intend to apply software engineering methodologies and tools to improve the CSE research software development.

Keywords: Software engineering, ATLAS, SeisSol, Bug report, Release planning, Reverse engineering, Scientific computing, High performance computing

1. Introduction

Software applications that are developed to serve computational science and engineering (CSE) research projects tend to be different from software applications developed for commercial purposes. The main goals of the CSE applications are evaluating the scientific computing approaches and generating the computation results [1, 2]. Therefore, most of the time and efforts are applied on how to achieve accurate and stable outcomes. This obviously shows that the correctness and efficiency of the software are the major concerns of the CSE applications developers. On the other hand, the comprehensibility, maintainability and extensibility of the software remains a minor issue of concerns for CSE developers, which are considered to be vital in commercial software development. For this reason, it is necessary to optimize the quality of the CSE research software since they have to be evolved for a long period of time to better assist the research [3]. In order to produce high-quality CSE software products, it is worthy to investigate how existing software engineering methodologies and tools could be used in the development of CSE software applications.

This paper presents an overview on the two ongoing projects from Munich Centre of Advanced Computing (MAC) - B2 [4] that apply software engineering methods and tools (e.g. release management, tools integration) in two different CSE applications in terms of scale and problem domains. The paper is organized as follows: Section 2

* *Email address:* naguib@in.tum.de, liya@in.tum.de (Hoda Naguib, Yang Li)
[1]Corresponding author

1877-0509
doi:10.1016/j.procs.2010.04.167

gives a brief description of the CSE research projects used in the case studies conducted, in which different software engineering challenges are identified. Section 3 presents a description of the proposed solutions that can tackle the problems presented in Section 2. Finally, the conclusion is presented in Section 4.

2. Project Descriptions and Problem Statements

There are differences when it comes to the comparison of development of CSE research software and commercial business software, such as: (1) having a large number of undiscovered requirements, due to the nature of CSE research process of exploring the unknowns, (2) the developers are focusing more on the research's computational output and performance rather than the software's comprehensibility, extensibility and maintainability, (3) most of the CSE developers don't necessary come from a software engineering background [2, 3].

The following subsections introduce the two case studies that are currently being conducted on the SeisSol project [5] and the ATLAS project [6]. A brief description of the two projects is given, in which problems are identified and then presented with respect to the projects different scales.

2.1. SeisSol : A small-scale CSE software project

SeisSol is a scientific software for the numerical simulation of seismic wave phenomena. Synthetic data sets and seismograms can be computed to support seismological research to understand observed data from the field and to determine the characteristics of sources and material structures [7]. The parallel computing is achieved by employing Message Passing Interface (MPI). It has been successfully run on SGI Altix 4700 supercomputers with ~ 2000 nodes.

The development of SeisSol started from 2006 and with a maximum of five developers. The main program language is FORTRAN 90 plus with few FORTRAN 77 subroutines. There are approximately 140,000 lines of executable code. Automated overnight compiling and validation with basic test cases are performed. So far, there are only a small number of users including the developers themselves. A preliminary user manual and related publications are available.

We can categorize SeisSol as a relatively small-scale software project, based on the development team size and the program size. The project started from sketching the key functionalities and the main functional modules. Then, these modules are mapped to codes directly by implementing various subroutines. Unfortunately, requirements are not elicited comprehensively at the early stage of the development. Undiscovered requirements emerge later on, and more effort is invested on patching them since it happens often that big code changes have to be made in order to implement the new features.

Project documentation is also incomplete. At the moment, only a user manual described how to get started with the SeisSol program and related scientific publications are available for references. Lack of documentation results in the following two chief disadvantages: (1) It can be a laborious work to review the code manually. Developers need to search for subroutines based on the FORTRAN code file names and be familiar with the main idea of each subroutine. Sometimes it can be even more difficult, since the implementation involves complicated numerical schemes and very often with parallel computing. (2) The system model and design are not visible or not clear. This can be a major obstacle to extending and maintaining the software.

In the SeisSol project, currently the main user group is the developer team. They are all using the developing version and there is no formal release process. However, stable versions that are validated by certain amount of reasonable test cases should be released, especially as the number of external users is increasing. Another problem in SeisSol is that bug reports and feature requests are not captured explicitly. Moreover, issues such as bug fixing, are not known or visible to all the developers. In this way, it becomes difficult for developers to keep themselves updated with information concerning bug reports such as status and the responsible developers.

2.2. ATLAS : A Large-scale CSE software project

ATLAS is a particle accelerator experiment at the Large Hadron Collider (LHC) located at CERN, Geneva, Switzerland. ATLAS is a high-precision particle detector used for detecting and classifying collisions of particles. The ATLAS team has developed a set of software and middleware tools which aim to fulfill three functions: (1) consolidation and filtering of raw data, (2) distribution of data on the Worldwide LHC Computing Grid (WLCG) [8] and (3) conducting scientific investigations on the data obtained such as simulation, reconstruction and analysis.

The main programming languages used are C++ and Python. The statistics form the source repository show that up to the year 2009 there are $O(10^7)$ lines of executable code. Moreover, within a two-week cycle, a new release candidate of ATLAS's software is released and deployed. Basically the release candidate consists of users requests, new features and bug fixes. The code of the release candidate is built in a nightly manner that it is further subjected to validation and testing procedures, in which failures and bugs are discovered.

There is a continuous growth in the number of ATLAS users and developers, which are about 2000 physicists and engineers from 204 institutions and 35 different countries. Therefore, we can conclude that ATLAS is a huge and globally distributed CSE research project that relies on a large software infrastructure, which subsequently involves a large number of developers. Moreover, software releases are usually planed and organized by the ATLAS Software team in a manual manner. This means that the release coordination, validation and development are challenging tasks that require constant maintenance and continuous improvements concerning the ATLAS project magnitude. In this case, considering options such as the use of automated release planning tool would be very beneficial.

The latest statistics shows that at least ninety bug reports are submitted per week by ATLAS developers and users. In addition, users face difficulties in submitting bug reports since it involves a lot of detailed technical parameters such as: version number, category and description. Moreover, ATLAS has an open bug repository without organization hierarchy, thus tracking and triaging are done manually. For that reason, not having a fully automated bug report life cycle increases the risk of having reports being neglected and delayed fixes [9, 10].

3. Proposed Solutions

This section presents some proposed solutions that we plan to implement for the two ongoing research projects ATLAS and SeisSol . These solutions attempt in overcoming the problems that were formerly mentioned in Section 2.

3.1. Customized solutions for SeisSol

Reverse Engineering. Due to the lack of documentation, reverse engineering is applied to explore the system and to learn the system architecture [11]. We start the reverse engineering with requirements elicitation by applying techniques such as interviewing developers, literature research, brainstorming, and source code understanding to discover the system requirements and record them into documentation. The requirements are considered from five main points: discretization, numerical schemes, parameters, output, and parallel computing. System analysis and design processes can work on these requirements. We examine how the system is designed and how to make improvements to it, especially on the extensibility and maintainabiliy.

Also a source code documentation generator tool, Doxygen [12], is employed to generate an on-line documentation of SeisSol. To date, the on-line source code documentation is automated generated whenever code changes are committed to the repository. In the documentation, Lists of modules and files bring a clear overview of the program and explanatory descriptions of subroutines and parameters are provided. Such a source code documentation is of great benefit to the implementation and the cooperation with other team members.

Release Management. To present a stable version with the core functionalities and additional features to users, a structured release management process should be considered which can also provide version control. In the SeisSol project, Subversion (SVN) [13] has been already successfully employed in the development to maintain the current version and the history of the source code. However, branching and tagging of the version control system are not yet applied, which can be used to plan and manage the release effectively.

Issue Tracking. To close the communication gap between developers, all of issues should be traceable and transparent to help the developers to be aware of the project state. It is easier for the collaboration since such a CSE research software is an inter-disciplinary subject with developers who has specific expert domain knowledge. Open issues can be also planned based on their priorities or broken down to smaller issues if they are too big [1].

3.2. Customized solutions for ATLAS

Automating the Release Planing Process. Developing an integrated release planning environment through which the manager can assign artifacts such as: issues, tasks and bug reports to specific releases or developers. Such an environment aids in automating ATLAS's software release planning process. Through this environment developers

can easily view all information concerning tasks assigned to them. In addition, developers can report their work progress in which managers can promptly view the status of the release plan, ultimately reducing communication gaps within ATLAS's software team. All of the above assists in the software development, release planning, and improving the overall productivity. More information about an existing CASE tool environment that can be applied on the ATLAS project is available in [14].

Enhancing the Bug Reporting Life Cycle. Our main goal is to enhance the current bug reporting life cycle available at ATLAS by introducing two main solutions, which are the automated bug triaging and linking releases with ATLAS's current bug reporting system [15].

- **Automated Bug Triaging:**
  The automated bug triaging solution intends to automate the process of submitting, assessing and assigning a bug report to be fixed. Therefore, making it easier for users to submit reports and for the developers to fix bugs. Furthermore, this solution is divided into four functionalities:
  - *Automatic collection of bug details:*
    This functionality intends to automatically gather technical bug related details (version number, IDE information, etc.) from the users environment settings at the time the bug occurred. After then the collected information can be added directly to the bug report without the user intervention.
  - *Duplication detection:*
    This functionality aims on discovering duplicates bug reports, i.e. preventing redundancy.
  - *Voting:*
    This functionality aims to provide a voting option that allows different users to state that they have experienced the same bug, which is quite useful in bug report quality evaluation. In addition, it will allow users to add different scenarios to a single bug report. This is done in case that more than one user have experienced the same bug within different circumstances.
  - *Developers assigning recommendation :*
    This functionality does not only allow the user to assign links between bug reports and developers, but it also recommends them. For example, it should recommend the developer who might be the most relevant to be assigned to solve a specific bug report. As a result the collaboration between developers will be enhanced and bug-fixing process will be simplified [9, 10].
- **Linking Releases with Bug Reports**
  This solution aims at developing a linking functionality to be added within ATLASs existing bug reporting system [15]. Through this linking functionality the release mangers will be able to assign a specific bug report to the concerning software release. Having such functionality will be of great assistance to both the release coordinator and the developers in handling the bug fixing and monitoring the release planning process. This will eventually enhance the quality of release planning and coordination process.

4. Conclusion

This paper illustrated two software engineering case studies that examine two different sized CSE research projects: a small-scale project SeisSol and a large-scale project ATLAS. Furthermore, we presented how software engineering methods and tools can be applied to enhance the software development of SeisSol and ATLAS. There are common problems between the two projects as well as specific ones. For the SeisSol project, we introduced the rationale and release management to solve existing development problems. For the ATLAS project, we suggested to apply automated release management and bug reporting in the project to ease the large amount of manual work.

5. Reference

[1] M. A. Heroux, J. M. Willenbring, Barely sufficient software engineering: 10 practices to improve your cse software, in: SECSE '09: Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering, IEEE Computer Society, Washington, DC, USA, 2009, pp. 15–21. doi:http://dx.doi.org/10.1109/SECSE.2009.5069157.

[2] J. C. Carver, R. P. Kendall, S. E. Squires, D. E. Post, Software development environments for scientific and engineering software: A series of case studies, in: ICSE '07: Proceedings of the 29th international conference on Software Engineering, IEEE Computer Society, Washington, DC, USA, 2007, pp. 550–559. doi:http://dx.doi.org/10.1109/ICSE.2007.77.
[3] J. Segal, Some challenges facing software engineers developing software for scientists, in: SECSE '09: Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering, IEEE Computer Society, Washington, DC, USA, 2009, pp. 9–14. doi:http://dx.doi.org/10.1109/SECSE.2009.5069156.
[4] Mac b2 project website.
URL http://www.mac.tum.de/wiki/index.php/Project_B2
[5] Seissol website.
URL http://www.geophysik.uni-muenchen.de/ kaeser/SeisSol/
[6] Atlas devloper's wiki.
URL https://twiki.cern.ch/twiki/bin/view/Atlas/WebHome
[7] M. Dumbser, M. Käser, J. de la Puente, Arbitrary High Order Finite Volume Schemes for Seismic Wave Propagation on Unstructured Meshes in 2D and 3D, Geophysical Journal International 171 (2) (2007) 665–694. doi:10.1111/j.1365-246X.2007.03421.x.
[8] Worldwide lhc computing grid (wlcg) website.
URL http://lcg.web.cern.ch/lcg/
[9] J. Anvik, L. Hiew, G. C. Murphy, Who should fix this bug?, in: ICSE '06: Proceedings of the 28th international conference on Software engineering, ACM, New York, NY, USA, 2006, pp. 361–370. doi:http://doi.acm.org/10.1145/1134285.1134336.
[10] J. Helming, M. Koegel, H. Naughton, Towards traceability from project management to system models, in: TEFSE '09: Proceedings of the 2009 ICSE Workshop on Traceability in Emerging Forms of Software Engineering, IEEE Computer Society, Washington, DC, USA, 2009, pp. 11–15. doi:http://dx.doi.org/10.1109/TEFSE.2009.5069576.
[11] K. Kreyman, D. L. Parnas, On documenting the requirements for computer programs based on models of physical phenomena (2002).
[12] D. van Heesch, Doxygen.
URL www.doxygen.org/
[13] Subversion (svn) website.
URL subversion.tigris.org/
[14] M. J.Helming, Unicase website.
URL http://unicase.org
[15] Savannah atlas bug-reporting system website.
URL https://savannah.cern.ch/