International Journal of Educational Sciences and Arts (IJESA)

Vol. (2), No. (1)

IJESA

February 2023

المجلة الدولية للعلوم التربوية والآداب

الإصدار (2)، العدد (1)

Impact of Applying Agile Practices to Bioinformatics Environment

Yasser Omar Elshamy

M.Sc. of Education Technology, Faculty of Education, Helwan University, Egypt

yasso.shamy@gmail.com

Fayez Ezzat Fergany

M.Sc. of Curricula and Teaching Methodology, Faculty of Education, Tanta University, Egypt

ferganyfayez1994@yahoo.com

Abstract:

The bioinformatics software development industry represents one of the fastest-growing fields. As a result of the lack of software engineering practices in the development and the complex nature of bioinformatics software development, there is a strong need for more agility in dealing with these challenges. The agile methods represent good development practices that rely on strong collaboration and automation to develop high-quality software within time and budget constraints through several iterations. This paper adopts agile principles, especially extreme programming (XP) practices to solve the common challenges that face the developers of bioinformatics software. The proposed agile practices can be used to facilitate and enhance the development processes which may increase the possibility of its success.

Keywords: Bioinformatics, Information Science, Software Development, Agile Practices.

1. Introduction

Bioinformatics is an interdisciplinary field of molecular biology, computer science theories, statistics, and mathematics that developing efficient algorithms to analyze, model, visualize and solve complex biological problems in plant, animal, and human including DNA, RNA sequences, protein sequences, and structures, microarray data, and next-generation sequencing data.

In April 2003, the Human Genome Project had completed by sequencing the 3 billion DNA letters in the human genome also it was planned for 15 years and cost $1 billion, now US company Illumina has announced it will begin shipping a system capable of sequencing the human genome for under $1,000 by producing around 600 giga bases of sequencing data per day.

There are three major international bioinformatics centers, NCBI, EBI, and ExPASy that collect, develop, and maintain hundreds of bioinformatics data and tools. The Bioinformatics field produces a large amount of complex data such as DNA, RNA, proteins, and other cellular molecules. A bioinformatician works to provide services to the scientific community in the form of databases and analytical tools. One of the challenges that will face the bioinformatics field over the next decade is the integration and presentation of the enormous and ever-expanding size data also it is necessary to integrate and present data from different views of the same system [1].

Since the completion of the human genome project, computer science tools have become indispensable in supporting: the modeling, analysis, integration, and visualization of large amounts of molecular data and advancing a major core of biological research. Bioinformatics has become one of the fastest-growing interdisciplinary scientific fields, combining Molecular Biology and Computer Science, among other disciplines. Many commercial and open-source tools of

bioinformatics have emerged, but they often lack transparency in that researchers end up dealing more with the complexity of the tools, rather than the scientific problems at hand [2].

The remainder of this paper is organized as follows. Section 2 gives a brief on bioinformatics software development agile methods. Section 3 represents the proposed agile practices. Finally in section 4 conclusions and future work inspiration.

2. Bioinformatics Software Development and Agile Methods

In bioinformatics software development, the primary stakeholders are biologists rather than computer scientists therefore it presents a unique situation for the field of software engineering, as it results in challenges and opportunities that are not typically found during the normal engineering process. Software engineering practices are still not of major importance in the bioinformatics field as the emphasis is on how to apply mathematics, and computer sciences to solve complex biological therefore there is still a large gap in understanding problems in bioinformatics software development [1].

2.1 Bioinformatics Software Development Challenges

Due to the complex and critical nature of bioinformatics software and its rapidly growing volume, there is a strong need to support bioinformatics professionals to develop maintainable and reliable software systems by applying software engineering practices. In addition, the bioinformatics domain is actually different in some aspects from the general software engineering community. First, in bioinformatics software development projects, the main driver of software requirements is to investigate sophisticated research questions rather than a more

generic business function, therefore the requirements will be complex, vague, and volatile, which presents an important risk for bioinformatics software efforts. Secondly, the strict budgets and schedule constraints of typical research projects proposed additional constraints for development; for example, the resources for appropriate testing, validation, and verification can be limited. Finally, bioinformatics developers, who may lack a formal software engineering background, are usually in a position to develop and maintain their own programs, i.e. there is a high proportion of end-user programmers in bioinformatics [4]. The lists of the main challenges in bioinformatics software development are [1]:

- Cross-disciplinary.
- Stakeholder heterogeneity.
- Lack of reusability.
- Project constraints.
- Documentation.
- Lack of Teamwork.

Cross-disciplinary: Bioinformatics is a cross-disciplinary field, it is one in which the two disciplines do not even speak remotely the same language.

Stakeholder heterogeneity: Stakeholders are biologists rather than computer scientists. Stakeholders may be more inclined to sacrifice program structure to get something that works.

Lack of reusability: Most bioinformaticians and computational biologists believe that good bioinformaticians build up their own toolbox, but the software development practices mostly surround the notion of "Don't reinvent the wheel" which essentially refers to the use of existing frameworks and take advantage of large existing projects like BioPython.

Project constraints: Tighter restraints on budget and timetables, as well as less time allotted for verifying and testing. Many bioinformaticians are doing the programming themselves, and have been left to their own devices in terms of software development and documentation.

Documentation: Do themselves and very limited if it exists.

Lack of Teamwork: Most bioinformaticians said that self-teaching was one of their main modes of software development process learning.

2.2 Bioinformatics Software Development Requirements

According to the nature of the bioinformatics fields, there are some requirements that should be taken into consideration to target the common challenges of this domain.

2.2.1 Approaches to Software Development

Rapid application development or prototyping is the best way to develop a tangible solution for the customer, but the drawback is that the prototype ends up being used as the actual system, which later results in problems. Future bioinformatics software developers should have complete knowledge of software engineering practices such as XP and related paradigms such as Test-Driven Development (TDD), and how to use the right mix for a successful project. Object-oriented concepts should be taught with rich examples and plenty of exercises [4]. In the experience reports on developing bioinformatics software by Kane [5] and Kendall [6], these reports emphasized that extreme programming and agile practices were indeed well suited to bioinformatics software development.

2.2.2 Importance of Documentation

In terms of software maintenance, writing an increased number of perceived comments and documentation is very helpful for the maintenance phase. Bioinformatics software is the most complex and constantly evolving one therefore, documentation is very important to developing software for bioinformatics software. The importance of documentation was stressed by the researchers of the Bioconductor project therefore documentation is considered a key practice to be strengthened in scientific software development [7].

2.2.3 Quality Assurance Practices

Bioinformatics research practice has critical implications for life sciences, and it is very important to have strong quality assurance (QA) practices such as code reviews and testing to ensure software quality. It would be very helpful to have a step-by-step tutorial about how to write a test case, as well as how to perform a code review. [4].

2.2.4 Software Evolution and Maintenance

The Bioinformatics field is still a very young domain, and software developed in this domain has not yet matured enough to be studied from an evolutionary perspective. As their applications move into legacy status, bioinformatics programmers need to understand more about the complex relationships among software size, complexity, and age, so that they can take preventive measures in advance [4].

2.2.5 Requirements Engineering

Managing requirements in the bioinformatics field is a challenging task. In bioinformatics, requirements cannot simply be “handed off” from the domain experts

to the degree that is possible in other disciplines. Close interaction and cooperation between domain scientists and professional developers are necessary in order to keep up with changing hypotheses, new algorithms, and new methods for handling vast quantities of data [8].

2.3Agile Methods

H. Frank Cervone, et al., 2010 [9] study displays the outgrowth of the agile software development movement. It states four core principles manifesto for agile software development as follows:

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

The most important agile methods include Scrum, extreme project management, adaptive project management, and dynamic project management method. H. Frank Cervone, et al., 2010 [9] say “the goal is to deliver a more suitable product more quickly than with traditional methods”.

2.3.1 Extreme Programming Practices (XP)

XP, originally explained by Kent Beck [16], XP is considered one of the first agile methodologies. XP practices consist of three cycles. Each cycle has methods.

- Planning game: planning game determines the scope of the next release and iterations. Progress tracking provides and enables through customer stories (requirements) written cards. User stories allocate to releases and iterations.

- Small releases: short iterations from one to four weeks lead to frequent releases from four to six months. From the whole view, each iteration/release makes sense. The next release has new customer thinking. Because requirements become clearer gradually and there is an option to change stories in any iteration small releases reduce risks.
- On-site customer: key user member in the team, he / her available full-time to answer questions through the real person who will use the system. Customer needs are reexamined all the time. Requirements problems are faced directly and handled professionally.
- Coding standards: standards are adopted by the whole team. Programmers write code according to rules ensuring communication through programming structure. Coding standards practice is connected to collective ownership, refactoring, pair programming, and continuous integration.
- Sustainable pace: target is to start work fresh within the official work time. No extra work and no overtime.
- Metaphor: there is a simple shared story that describes how the whole system works to guide all development entities.
- Continuous integration: many times a day integrating and building the system, every time a task must be completed. Integration machine methodology has one set of changes integrated at a time.
- Collective ownership: If programmers decide to improve, they can manipulate any code anywhere at any time in the system. The collective ownership concept is to reduce the technical risk in case of a programmer leaves the organization.
- Testing: Test cases are written before the code is written. Testing is an essential part of the coding process.

- Refactoring: programmers tune the system structure without changing its behavior. They remove duplication and improve communication and simplicity. Code improvement is a must.
- Simple design: Collective ownership and refactoring strongly support simplification. Simple design improves program structure understanding.
- Pair programming (PP)**:** all code is written within pairs and changes according to tasks and teamwork expertise. Programmers prefer PP because it helps them to discover bugs earlier, the team accepts decisions of pairs, and increases productivity and quality.

2.3.2 Test-Driven Development (TDD)

Test-Driven Development (TDD) is an advanced technique for developing software that guides software development by writing tests. Antti Hanhineva [14] illustrates the following TDD definitions:

- TDD is an agile practice where the tests are written before the actual program code.
- TDD is a technical enabler for increasing agility at the developer and product project levels. Existing empirical literature on TDD has demonstrated increased productivity and more robust code, among other important benefits. TDD is an incremental process. First, a test is added and the code of the test is written. If the test is passed, then the code is refactored.
- Refactoring is a process of making changes (tuning) to existing and working code without changing its external behavior, i.e., the code is altered for the purposes of comments, simplification, or other quality aspects. This cycle is repeated until all of the functionality is implemented.

TDD identifies several potential benefits as follows:

- Developer confidence.
- Efficient refactoring.
- Fast debugging.
- Software improvement.
- Safety changes.
- Up-to-date code documentation.
- Help developers avoid over-engineering by setting a limit on what needs to be implemented.

2.3.3 Scrum

Scrum is a part of the Agile umbrella that was introduced in 1995 by Ken Schwaber and Jeff Sutherland. Scrum is an agile software development framework that is widely used to achieve agility, and incremental and iterative development in the software development cycle. Scrum is focused on project management that can manage and control software development while XP focused on software development of rapidly changing requirements. The Scrum activities are [17]:

- Preparing product backlog.
- Sprint planning meeting and preparing sprint backlog.
- Sprint.
- Daily Scrum.
- Sprint review and presenting an increment.
- Spring retrospective.

3. The Proposed Agile Practices for Bioinformatics Software Development

The goal of the paper is to propose agile practices for enhancing the development of bioinformatics software and overcoming the bioinformatics software development challenges in spite of excluding the agile methodology in others [1]. M. M. Muller and W. F. Tichy, et al., 2001 [11] saw that the lightweight nature of agile methods affords a lot of flexibility to the development process but makes agile methods difficult to implement in a disciplined manner without coaching.

An undisciplined application of agile methods leads to a "patch and go" attitude. Agile methods are commonly used in the development of scientific software (e.g., [12], [13]).

The proposed agile practices will be enhanced to use and utilize the following agile methods to overcome the bioinformatics software development challenges as shown in Figure 3:

- Extreme Programming (XP).
- Test Driven Development (TDD)
- Scrum

3.1 Extreme Programming (XP)

XP has a powerful cycle that guides standard software development and continues integrations. XP practices can overcome the lack of teamwork, project constraints, lack of reusability and can deal with stakeholder heterogeneity.

3.2 Test-Driven Development (TDD)

TDD is an advanced technique that drives the design of the software by using unit tests. In figure 3, TDD is used to overcome the lack of documentation because of the unit tests act as self-documentation.

3.3 Scrum

Scrum is focused on project management skills. In figure 3, Scrum is used to manage the cross-disciplinary, stakeholder heterogeneity, and lack of teamwork.

4. Conclusion and Future Work

This paper is trying to apply the agile methodology for Bioinformatics Software Development (BSD). As was believed agile has some methods and features that must be used in software development bioinformatics (BSD). Bioinformatics science and agile technique are discussed to be synchronized especially with extreme programming practices. Test-driven development (TDD) appears as a centralized core face. TTD discovers the implicitly hidden knowledge of bioinformatics developers. TDD interpreted this knowledge on specific test code cases. Scrum can enhance the management skills of bioinformatics projects in spite of finding a cross-disciplinary with different background.

The contribution of this paper demonstrates and gives some of the software engineering logic that agile has the flexibility to be adopted with specific science like bioinformatics. There is a need to be improved in terms of quality assurance, also proposed framework needs to have experimented, and finally real software development bioinformatics projects should be included in future works.

References

[1] Dhawal Verma, Jon Gesell, Harvey Siy, and Mansour Zand, "Lack of Software Engineering Practices in the Development of Bioinformatics Software.", In ICCGI 2013, The Eighth International Multi-Conference on Computing in the Global Information Technology, pp. 57-62. 2013.

[2] Chilana, Parmit K., Carole L. Palmer, and Andrew J. Ko., "Comparing bioinformatics software development by computer scientists and biologists: an exploratory study", In Software Engineering for Computational Science and Engineering, 2009.SECSE'09. ICSE Workshop on, pp. 72-79. IEEE, 2009.

[3] Chen, Hsinchun, Sherrilynne S. Fuller, Carol Friedman, and William Hersh, "Medical informatics: knowledge management and data mining in biomedicine", Vol. 8, Springer, 2006.

[4] Umarji, Medha, Carolyn Seaman, AkifGünes Koru, and Hongfang Liu, "Software engineering education for bioinformatics", In Software Engineering Education and Training, 2009.CSEET'09. 22nd Conference on, pp. 216-223. IEEE, 2009.

[5] Kane, David, "Introducing agile development into bioinformatics: an experience report", In Agile Development Conference, 2003, ADC 2003, pp. 132-139. IEEE, 2003.

[6] Kendall, Richard, Jeffrey C. Carver, David Fisher, Dale Henderson, Andrew Mark, Douglass Post, Clifford E. Rhoades, and Susan Squires, "Development of a weather forecasting code: A case study", Software, IEEE 25, No. 4 (2008): 59-65.

[7] Gentleman, Robert C., Vincent J. Carey, Douglas M. Bates, Ben Bolstad, Marcel Dettling, Sandrine Dudoit, Byron Ellis et al., "Bioconductor: open software development for computational biology and bioinformatics", Genome biology 5, no. 10 (2004): R80.

[8] Letondal, Catherine, and Wendy E. Mackay. "Participatory programming and the scope of mutual responsibility: balancing scientific, design and software commitment." In Proceedings of the eighth conference on Participatory design: Artful integration: interweaving media, materials and practices-Volume 1, pp. 31-41. ACM, 2004.

[9] H. Frank Cervone, Understanding agile project management methods using Scrum

[10] Cory Foy, Figure of Extreme Practices (XP), www.xpprogrmming.com. foyc@cornetdesign.com. http://www.cornetdesign.com. Downloaded on 30.10.2021 7:00 PM.

[11] M. M. Muller and W. F. Tichy, “Case study: Extreme programming in a university environment”, Proceedings of the International Conference on Software Engineering, pp. 537–544. (ICSE 01), 2001.

[12] O. Chirouze, D. Cleary, and G. G. Mitchell, “A software methodology for applied research: extreme researching”, Software: Practice and Experiences, vol. 35, no. 15, pp. 1441–1454, 2005.

[13] W. A. Wood and W. L. Kleb, “Exploring XP for scientific research”, IEEE Software, vol. 20, no. 3, pp. 30–36, 2003.

[14] Antti Hanhineva ElbitOy, JuhoJäälinoja, IMPROVING BUSINESS AGILITY THROUGH TECHNICAL SOLUTIONS: A Case Study on Test-Driven Development in Mobile Software Development. Pekka Abrahamsson, VTT Technical Research Centre of Finland. Downloaded on: 06.12.2014 09:08 P.M. http://agile.vtt.fi/docs/publications/2005/2005_business_quality_ifip.pdf. Agile.vtt.fi, 2005

[15] David W Kane, Moses M Hohman, Ethan G Cerami, Michael W McCormick, Karl F Kuhlmman and Jeff A Byrd, Agile methods in biomedical software development: a multi-siteexperience report, http://www.biomedcentral.com/1471-2105/7/273/, Doi: 10.1186/1471-2105-7-273. Downloaded on: 29.11.2014 08:25 P.M. *BMC Bioinformatics* 2006.

[16] Beck, Kent. Extreme programming explained: embrace change. Addison-Wesley Professional, 2000.

[17] Nagy Ramadan Darwish, "Enhancements in Scrum Framework using Extreme Programming Practices", International Journal of Intelligent Computing and Information Sciences (IJICIS), Ain Shams University, Vol. 14 No. 2, Page: 53-67, April 2014.