EPiC Series in Computing

Volume 58, 2019, Pages 1–7

Proceedings of 34th International Conference on Computers and Their Applications


# Planning Computational Biology Projects Using Agile Approach

Tamer Aldwairi[1,2]
[1]Mississippi State University
Starkville, MS 39762, USA
[2]Ursinus College
601 E. Main Street
Collegeville, PA 19426, USA
taa70email@gmail.com

**Abstract**

Recent advancement in the biological sciences field led to an increase in the development of large biological software research projects. These projects are complex and interconnected and have proven to be hard to manage. In many cases, these projects are not completed within their deadlines and fail to provide the users with a reliable piece of software that can be managed and maintained in the future. This observation was also confirmed by scientists working in other research and scientific fields. It is well known that the non-deterministic nature of science requires it to be always evolving and changing. To meet such requirements using conventional software engineering practices will be a very hard goal to achieve. For this reason, I suggest the use of agile approaches such as Scrum, Kanban, and extreme programming, as an attractive choice to many developers working in many academic and scientific fields especially in areas such as computational biology. In this paper, I will discuss some biological software systems that used agile approaches in developing or enhancing their projects and I will look into the benefits gained from adopting those approaches and how it can benefit the outcome of many scientific projects.

## 1 Introduction

Lately, software development of scientific research projects has been the concern of many researchers. This can be seen more noticeable in an interdisciplinary type of fields that require computer intervention such as programming, simulation and a substantial amount of computations. These fields emerge through combining techniques, methods, and approaches from computer science, physics, chemistry, applied mathematics, statistics, and biology. That is why developing software products within fields distinguished as having an interdisciplinary environment is very challenging and

demanding and requires a very high level of communication among researchers [3]. These days computational biology projects are growing rapidly both in volume and complexity at a very fast pace. The amount of data that we need to process is becoming excessively large. This is mainly due to the advancement in technologies addressing biological problems such as next-generation sequencing and micro-arrays. These rapid changes increased the amount of biological data that we are required to store, process and utilize in our research [8]. The problem of handling such a large and complex data with so many features came to be known later as big data. Those complex systems can be modeled only through the organized set of approaches, models, and rules embedded in software engineering practices to govern the rapid changing aspect of science. Using such methods would help us in minimizing the number of deficiencies, bugs, and refinements required for each project as it is evolving. One important aspect of these methods is project management and planning. In this paper, I will discuss how project planning using agile methods can be used to improve the outcome of biological projects.

# 2 Background on agile programming

The agile manifesto was introduced in 2001 [9], but agile methods and practices were used a long time before the release of the manifesto. Agile programming incorporates many methods. Among the most common are Extreme Programming (XP) [10], Scrum [11], Kanpan [14] and Feature-driven development [12]. These methods share basic concepts but are still characterized by certain distinct features that make it more suitable to use in certain situations more than the others. Some of the features that differentiate them are the way the requirements are gathered, the team composition, their roles and responsibilities and the type of documents used. For example when comparing XP and Scrum we find that the size of a team in XP usually ranges between 3 to 16 members per team while in Scrum they usually range between 5 to 9 members per team. In XP we usually have one team per project while in Scrum we could have more than one team working on one project [11, 12]. One of the main distinct advantage for using agile methodologies in comparison to other classical development approaches is their ability to easily adapt to changing requirements where most classical software engineering approaches require a detailed description of the requirements and are less tolerant to sudden changes in requirements.

# 3 Motivation

In the computational biology field, many software projects run through websites and require access to very large databases. The programs and tools used on those sites have to process many requests and transactions throughout the day. Their DB's are increasing in volume on a weekly or even on a daily basis and are growing at an exponential rate with the addition of more sequences and the continuous emerging of new sequencing technologies. In addition, it is expected that in the near future some of these DB’s and sites might be integrated together [1]. This will provide the users with much more abundant data throughout their search but at the same time will increase the complexity of the whole system. The integration of such DB’s will increase the number of interactions between these components resulting in higher complexity. Data management of such systems will depend on the system’s ability to store and retrieve a large amount of data within a specific time and the system’s ability to interact with outside well-known data sources within the same field [7]. Such systems will be very hard to manage, especially if they were built without software engineering practices in mind. This brings us to a fact that was demonstrated in a recent survey study. The study showed that software engineering practices are rarely mentioned or used in computational biology projects [2].

A good example of this can be seen in many computational biology websites. Where a certain tool was developed for research purposes and after the tool resulted in publishing a paper, it became obsolete. In that sense, when other researchers try to use the tool, they found that it requires them to change the format of the data into a rare old format or that the tool is very slow in processing requests and in some other cases the tool was not updated to support the latest Operating system. I also noticed especially in the computational biology field that these programs process large amounts of data to be able to show results. This requires these tools to be designed using the best and most efficient algorithms to be able to perform a large number of requests at a very fast pace. I noticed that the majority of online biological research tools lack in that area.

Planning for such large projects is of significant importance since these projects require integration of different aspects of software techniques, continuous update of data and most notably the ability to integrate with other computational biology projects in the future [1]. For such reasons and based on the surveys and studies that discussed the minimal use of software engineering practices in computational biology projects [2]. I explored the availability of computational biology projects that incorporated software engineering approaches throughout their development or that were enhanced through adding new functionality to the project through employing software engineering techniques or practices and had resulted in improvement of the whole system. I found that some computational biology projects have used software engineering approaches that resulted in great benefits for both the developer and the stakeholder. I will discuss those projects in section 5 of this paper but before that, I need to address some of the challenges that limited the use of software engineering practices in computational biology projects.

# 4 Challenges of incorporating SE practices in CB

Using SE practices in developing CB projects is still new to the biology field. Only a few projects used or tried using SE approaches and techniques during the development of their software systems. Those projects reported positive feedback thorough out all the phases of development of their systems. Some of the benefits that these projects gained included reduction of time and cost and higher customer satisfaction of the product. If this was the case, why were not SE practices applied on a larger scale in the development of biological software applications? There are a number of reasons but before that, I need to point out that planning and project management of scientific applications are usually different than their counterparts in business applications. When we add to that managing CB projects, the concept becomes more distinct since CB projects incorporate different aspects from multiple disciplines. One thing we know for sure is that when SE methods were first developed they were not designed with scientific applications (SA) in mind. Below I discuss some of the reasons and challenges that I believe had limited the use of SE in the development of CB projects.

The first challenge that appears to a software engineer is the unpredictable changes that appear in the work settings. Requirements in a scientific environment are usually random. There might be a goal that needs to be reached but the means for achieving such a goal are not clearly stated, meaning that they can be achieved in many different ways or using a number of different techniques that will lead to a certain event. This event can simply be a new discovery in that area that will contribute to the advancement of science in general. Planning in such situations can be a complex task and is totally different than planning for a business model.

The second challenge is the lack of a clear definition of a stakeholder role in the system. For example, in an academic setting, the stakeholder role cannot be easily defined. In certain cases, the

stakeholder could be the major adviser of a graduate student or the student himself, if the program he/she is developing is part of his/her own research [4] or the government or some private/public institution like the university or the combination of all the previous examples. In other cases, the stakeholder could be a research group facility interested in using the software as an application to aid in the advancement of their research. In such a case the stakeholder role and his level of involvement in the project are on a totally different scale. Of course, as you can see this argument applies to all the other parts of the SE life cycle such as planning, analysis, design, testing, implementation, release and maintenance. All those parts and even certain sub-parts have to be redefined to get a clear view of how those roles interact with the system.

The third challenge which is usually more obvious than the others is the necessity of frequent releases during the development of the software. In scientific research, it is important to have a working piece of software as soon as possible. Since research cannot be postponed until a full-fledged product is available for use. Planning and managing releases in such settings can be challenging and might require a different form of management and planning. Especially compared to the ones used in a business setting. Since funding could be limited to shorter periods of time, planning time frames and iterations for a project might be shorter than usual.

The fourth challenge and probably the one that has the highest impact of all is the lack of historical data regarding efforts, cost and time estimations. Since the use of SE in scientific projects is still new there are not enough reliable data from the past to do perform good project time and cost estimates. Since previous projects that share similar characteristics can be used to give good estimates regarding effort, measured in man-hours, needed to complete the project. This will make the job of planning and time scheduling a difficult task. Another important aspect regarding this point is the renewable nature of scientific research that does not allow for the accumulation of historical data. The availability of historical data might be limited to a scientific research project because we are always trying to produce something new. The need to develop a system that is already available is rare since most research and scientific software systems are available for free when used for non-commercial purposes.

# 5 Agile programming as a solution

Science is an exploratory process in which a certain hypothesis is tested and is either accepted or rejected based on certain evidence or results that support that hypothesis [5]. In a scientific environment, the requirements are not clear at the beginning of the project. In addition, the requirements keep changing rapidly [6]. The scientific environment could be a Ph.D. student working on a small scientific project and getting the requirements from his adviser at each meeting. These requirements could be changing based on how the project proceeds. An SE approach that might prove to be useful in solving such a situation is the employment of agile programming techniques to accommodate the continuous change of requirements. Agile methods and scientific applications share a certain characteristic that is common to both of them, they are both emerging processes. One of the twelve principles of agile methods is that it should “welcome changing requirement even late in development” [13]. Because of that when agile methods are used in developing software I do not worry a lot about future design. With this in mind, I can start developing the project and not be burdened regarding the emergence of sudden new requirements. Such a method could prove to be very useful, especially considering that it satisfies the unyielding change needed in a scientific environment.

Another situation that demonstrates how effective the agile approach can be is the academic settings of research. Where for example a graduate student writes code for his research that is only understood by the student himself. After the student leaves the department he used to work in, the program becomes

useless and obsolete. In the best scenario, it can be used as a black box without any modifications or improvements. Agile methods could be suitable in such a situation since it includes a number of techniques like pair programming, which when used in developing CB projects proved to be very useful and advantageous. That is true because in any academic environment it is crucial that each software component is understood by more than one individual. In case a single individual had to leave the project, the development can continue without many interruptions [4].

# 6 Incorporating SE in CB projects

To our knowledge, the use of SE practices in CB has not been thoroughly studied. It seems that the incorporation of such practices might be beneficial to such a field. In fact, some researchers have suggested incorporating some SE courses within the CB curriculum [2]. Of course, SE approaches were originally designed to work with business fields so using it within the CB field will require certain changes to the common SE rules and practices that are usually used in business. These changes will reflect something probably similar to the use of SE in a scientific environment. Such changes must be dependent on the scientific field of interest and will be tailored toward addressing the problems that usually occur in that specific field, but this is beyond the scope of this paper.

Recent studies have shown that the use of agile programming methods in the development of CB software can be very beneficial. For example, in [6] agile methods where used incrementally for software development at the National Cancer Institute in Bethesda, Maryland. The paper describes how successful this combined approach was and suggests incorporating such methods for the development of scientific applications. The project was originally developed using proof-of-concept and the agile approach was introduced and used later. When using the agile methods the team decided to adopt an incremental approach for iterations and a CVS (concurrent versions system) for configuration management. Managing tasks and requirements were initially done informally through e-mail and conversations. Later they introduced the planning game approach were they assigned story points to estimate the amount of effort needed to complete a task. They measured the velocity in developing the project by counting the finished points during each iteration. From their experience in this project, the authors believe that agile methods are a very good match for developing software projects inspired by science.

In [5] agile methods were used to plan, build, and test projects at six biomedical software development organizations. One fundamental challenge that faced the projects during the planning phase was making the development teams accustomed to accepting the idea of continuous change in the requirements and activities of the projects. Team and developers who had worked in the past with normal software projects will find the idea of changing requirements during a software project very hard to digest. In all six projects, an iteration planning approach was used. In each one of the iterations, the team's goal was to reach a stable point of completion. After each iteration, the project progress was compared against an estimated time plan to find out if the teams were ahead or behind the time schedule. All the groups used backlog as their main planning tool. Planning new iterations was done through selecting features that needed to be done and estimating the effort needed to complete them. In five of the projects, customers placed more emphasis on frequent release dates than fixed release dates. In only one project, fixed release dates were more important because the project was tied with a larger product release. The authors of the paper believe that the benefits of adopting the agile methodology can be extended to biomedical informatics and proved to be very helpful since it suits the exploratory nature of science.

Another case that supports our claim is the Chaste - Cancer, Heart, and Soft Tissue Environment - project. This is a computational biology software library that the authors of [4] experimented on using agile methods. The team continued on using the agile approaches after the experiment was concluded with some modifications to the agile process. The use of agile methods was very beneficial in producing high-quality software that is adaptable to the software requirements. Planning was done through release and iteration planning, where the users and the customers agree on the stories that should be in each release. Each release is then divided into iterations and the stories to be implemented are chosen from the release plan. Then the stories get unpacked into smaller tasks and a more accurate estimation effort will be required at this stage. Some of the advantages that the paper addressed when using this method is the ease of integrating a new member into the project or substituting someone who has left the project. This is necessary because in academia it is common for researchers to leave ongoing projects after they finish their contracts with the institution in which they worked. This can make effort planning a lot easier than when using other methods.

# 7 Conclusion

In this paper, I demonstrated that incorporating SE practices and methods in the development CB project might turn out to have a positive impact on the CB software committee. These projects could grow as large and complex as commercial enterprise business projects. That is why incorporating SE methods might be a necessity to save time, cost and effort in their development. I also call attention to a recent survey study [2] that showed the lack of using SE practices in the development of CB software applications and identified the areas needed to bridge the gap between bioinformatics, software systems, and SE practices. I illustrated that most of the current SE methods are not suitable for the continuously changing nature of science and that these methods need to be reconstructed from a scientific perspective rather from a business point of view. I also have observed that agile methods are the most suitable among all SE methods to apply to scientific projects, especially the ones related to CB. The reason for that lies in their ability to adapt to continuous changes in the environment. I support our claim with a number of studies [4, 5 and 6] that show the advantages of using agile practices in planning and developing CB projects. Based on the results of these studies I came to the conclusion that the advantages of using agile programming in developing CB projects outweighs both the challenges and disadvantages of developing these software projects without any use of SE methods.

# Future Work

Our future work will include expanding this study to include other phases of the software development life cycle (SDLC). This will include analysis, design, testing, and implementation for the projects discussed in the paper.

# References

[1] C. Goble and R. Stevens, “State of the nation in data integration for bioinformatics”, Journal of Biomedical Informatics, Volume 41, Issue 5, October 2008, Pages 687-693.

[2] M. Umarji, C. Seaman, A. Koru, H. Liu, "Software Engineering Education for Bioinformatics," Software Engineering Education and Training, 2009. CSEET '09. 22nd Conference on , vol., no., pp.216-223, 17-20 Feb. 2009.

[3] J. Barker and J. Thornton., "Software engineering challenges in bioinformatics," Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference, vol., no., pp. 12- 15, 23-28 May 2004

[4] Pitt-Francis, Joe; Bernabeu, Miguel O.; Cooper, Jonathan; Garny, Alan; “Chaste: using agile programming techniques to develop computational biology software,“ Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, Vol. 366, No. 1878. (13 September 2008), pp. 3111-3136.

[5] D.W. Kane, M.M. Hohman, E. G. Cerami, M. W. McCormick, K.F.Kuhlmman, J. A. Byrd, “Agile methods in biomedical software development: a multi-site experience report”,BMC Bioinformatics, Vol. 7, No. 1. (1 May 2006), pp. 1-12.

[6] D Kane, "Introducing agile development into bioinformatics: an experience report," Agile Development Conference, 2003. ADC 2003. Proceedings of the , vol., no., pp. 132- 139, 25-28 June 2003.

[7] Jordan, C.; Stanzione, D.; Ware, D.; Lu, J.; Noutsos, C.; , "Comprehensive data infrastructure for plant bioinformatics," Cluster Computing Workshops and Posters (CLUSTER WORKSHOPS), 2010 IEEE International Conference on , vol., no., pp.1-5, 20-24 Sept. 2010

[8] Takahashi, K.; Yugi, K.; Hashimoto, K.; Yamada, Y.; Pickett, C.J.F.; Tomita, M.; , "Computational challenges in cell simulation: a software engineering approach," Intelligent Systems, IEEE , vol.17, no.5, pp. 64- 71, Sep/Oct 2002

[9] Beck, Kent; et al. (2001). "Manifesto for Agile Software Development". Agile Alliance. (Accessed 2018-10-27).

[10] K. Beck, Extreme Programming Explained: Embrace Change, Addison-Wesley, Boston, 1999.

[11] Ken Schwaber and Mike Beedle, Agile Software Development with Scrum, ISBN 978-0130676344, Prentice-Hall, 2001.

[12] Palmer, SR., Felsing, JM. “A Practical Guide to Feature Driven Development”, Prentice Hall, 2002.

[13] Beck, Kent; et al. (2001). "Principles behind the Agile Manifesto". Agile Alliance. (Accessed 2018-10-27).

[14] Reinertsen, Donald (May 2009).” The Principles of Product Development Flow: Second Generation Lean Product Development”. Celeritas Publishing.