Computer Science 130, May 13th 2019 Debate Essay

Michael Wu
UCLA

1 Introduction

Recently the Event Horizon Telescope (EHT) has been in the news because it captured the first image of a black hole. This project involved a coordinated research effort from scientists across the globe. It also involved large amounts of computing power. Researchers ran High Performance Computing (HPC) software to collect and process the data used to render the image of the black hole. Our debate topic will address the topic of whether Constructive Cost Model (COCOMO) II, a software cost modeling method, would have been a good choice for modeling the costs for the HPC software developed for the EHT. Although there are many pros and cons for this choice, the age of the COCOMO II model and the presence of alternative models that are newer and better means that we would not recommend its use.

2 Background

The EHT is an array of radio dishes across the Earth that work together to create a virtual telescope. This virtual telescope has a very wide effective diameter, allowing it to generate images of objects far away. Currently there are 11 observatories participating in the EHT array [4]. All the observatories collect large volumes of data from their radio dishes, which are then stored and processed in order to render images. In some cases the data must be physically shipped on hard drives in order to gather it for processing, since some of the EHT observatories are located in remote regions [4]. HPC software is necessary for both the collection and processing of the data, since there is such a large volume of it.

One main component of the EHT software is the data collection backend, which allows each observatory to collect the necessary data for imaging. This process involves a software package called Distributed FX (DiFX) which helps correlate the data collected by each dish. DiFX runs on "clusters of more than 1000 compute cores at each site" which communicate with Mark 6 playback units that play back the data at rates of up to 16Gbps [4]. In addition to this, the main HPC algorithm used to render the image of a black hole was developed by researchers at MIT, Harvard, and Google. It's known as CHIRP (Continuous High-resolution Image Reconstruction using Patch priors), and it borrows statistical imaging techniques from the field of computer vision [3]. These components use HPC because they need to operate in parallel at high speeds in order to get good results. While developing the software to implement this algorithm and the data collection backend, researchers could have considered using cost modeling techniques to understand their development costs.

One such model is COCOMO II, a method that dates back to 2000 which was developed at USC [1]. It is a parametric model for software cost estimation, meaning that project managers can input information about various features such as personnel capability, product reliability, personnel experience, and functionality requirements in order to predict how much effort a software project will take [1]. Using data from previous software projects, the COCOMO II model includes multipliers for each of these factors that weigh how important they are. Eventually after numerous calculations the model outputs the required number of person-months to complete the software. We will examine the various arguments for and against the choice to use COCOMO II.

3 Pros

First we should address why software cost estimation must be done at all. Couldn't developers just begin working without doing cost estimation? It doesn't affect the implementation of the software and adds additional overhead. Well, cost estimation should be done since it allows developers to make the most of their budget. Although it is not a commercial project, the EHT still requires funds to operate. These funds come from research grants, and it would be wise to do cost modeling so that these funds are not wasted. Cost modeling also allows developers to understand what can be realistically achieved so that projects finish on time. It structures the development process so teams know what is left to be done and how long it will take. With this information, managers can make decisions such as adding more manpower or reducing the scope of the software in order to finish on time [5].

COCOMO II appears to be an attractive option for cost modeling due to its ease of access and the research behind it. It is publicly available and free to use. Additionally, USC has made available open source software that assists with performing the calculations in the model. Other models may not have similar levels of support since they might be more obscure or require payment to use. COCOMO II has also been studied and refined over a long period of time, since it is a revised version of the initial COCOMO model that was developed in 1981 [1]. It uses data from real world projects to come up with its estimations, so it seems like a reasonable choice.

Furthermore, COCOMO II is flexible and can accommodate various software types. It can use standard lines of code (SLOC) or function points to make estimations [1]. SLOC refers to the size of the code required, adjusted for language specific factors, while function points refer to the necessary functionality of the software. COCOMO II also has different methods for estimating development cost during different stages of the development lifecycle. There is an early design model, application composition model, reuse model, and post architecture model [9]. The early design model deals with software after the requirements stage, and uses function points for estimation. The application composition model deals with software that is built by putting together existing components along with database programming and scripting. The reuse model deals with reusing existing code and integrating it into the project. The post architecture model deals with estimation after a detailed architecture is formulated. With all these options, COCOMO II can handle a wide range of development efforts. It also allows for estimations to be refined as development progresses for increased accuracy. Lastly, since it is a formulaic method it will lead to repeatable results. This is beneficial since managers can predict how adding or removing features will affect their development costs.

Specifically for the EHT project, COCOMO II would be a good fit due to the nature of the EHT software. The EHT project began in the 2000s when COCOMO II was still fairly new, so the real world data used to calibrate COCOMO II would still be accurate. Also, the EHT software is developed over a longer time frame so long term planning would work. It is not a business application that uses agile and pushes a new build every two weeks. Both the data collection software and the data processing software needs to have its main functionality before release. Lastly, since the EHT software is mainly concerned with performance, this reduces many of the unknowns that come with business applications. The EHT developers do not have to worry as much about safety, security, user experience, and changing business needs. This makes COCOMO II more predictable and a better way to assess development cost.

4 Cons

Although COCOMO II may be a reasonable choice for cost modeling for the EHT, there could be other options that are better. COCOMO II is somewhat old, since it was created almost 20 years ago. During this time there may have been many changes in how software is made, so the model might not fit well anymore. New management methods, available technology, language constructs, and design standards may have affected development costs. Additionally, COCOMO II appears to have been developed with government and business projects in mind, as shown by the sponsors that funded its development:

> Aerospace, Air Force Cost Analysis Agency, Allied Signal, AT&T, Bellcore, EDS, Raytheon E-Systems, GDE Systems, Hughes, IDA, JPL, Litton, Lockheed Martin, Loral, MCC, MDAC, Motorola, Northrop Grumman, Rational, Rockwell, SAIC, SEI, SPC, Sun, TI, TRW, USAF Rome Lab, US Army Research Labs, Xerox [1].

In *Software Engineering*, Sommerville even notes that "the practical application of algorithmic cost modeling has been limited to a relatively small number of large companies, mostly working in defense and aerospace systems engineering" [9]. This is a very specific category that the EHT software may not fit into, as it an academic project that involves HPC. COCOMO II's lack of widespread adoption in the software engineering industry indicates that it does not provide clear, proven benefits in the development process.

Additionally, COCOMO II is hard to understand and use correctly. It relies on parameters such as personnel capability, product reliability, and personnel experience, but these are all subjective qualities. How does somebody conclude that they have a highly capable team? People tend to not know what they are deficient at, since they may not know enough about what they don't know. Even parameters such as SLOC or function points may be hard to accurately assess, since functional requirements may have varying degrees of difficulty and lines of code may not directly correlate to code complexity. Because COCOMO II is not widely used, developers will be unfamiliar with it. From personal experience I have not seen COCOMO II used in industry. It seems to be a niche technique that has been researched at only one institution, USC.

Also, the EHT project does not work well with COCOMO II since it is a widespread academic project that uses HPC. There are many teams working on the project across the world, so it may be hard to communicate about progress and assess every component of the project. Qualities such as team cohesion would be hard to estimate, so the inputs to the COCOMO II model would be inaccurate. Also organizational structure of the EHT project may not be as clearly defined as in private company, and academics tend to have more responsibilities than just producing code. This means that it would be hard to come up with good parameter values for estimation. While investigating cost modeling for other HPC applications, a study by Miller et al. [6] concludes that "COCOMO II is not directly applicable to the investigated HPC projects" and explains why:

> The main reason for COCOMO II not being directly applicable to the observed HPC projects lies in the inaccuracies in the parameter ratings. These occurred due to multiple reasons: First, the early project stage when the parameter rating needs to be applied to maintain feasibility and sophisticated resource management is error prone since the expert rating is limited by the available information [7]. Additionally, Pendharkar et al. [8] states that the rating of an expert, such as a project manager, changes over the project life cycle and the uncertainties in the characteristics of a project decreases with the proceeding life cycle [2]. Second, the parameters describing the experience of the developers (AEXP, LTEX and PEXP), are not only subjective but the reference group (nominal rating) is typically unknown.

Lastly, there is no reason to choose COCOMO II over alternative cost modeling methods. A study by Toka and Turetken [10] evaluated COCOMO II against other parametric cost models: SEER-SEM by Galorath, SLIM-Estimate by QSM, and TruePlanning by Price-S. They found that COCOMO II "scored the lowest in terms of effort estimation accuracy" in comparison to these other methods. These other cost models are all backed by companies who assist in the cost estimation process. Thus they are easier to use and they impose less management overhead.

Alternatively, development planning can occur in an informal manner. Since "estimates created early in a project are inherently inaccurate" [5], there is no point in trying to put so much detail into the estimations. A rough guess by the developers who are working on the project would suffice. This method is how agile methods tend to work, and from personal experience this worked fairly well in my industry job.

5 Recommendation

Overall I would recommend that COCOMO II should not be used for modeling the HPC software used in the EHT. Although software cost estimation is important and provides valuable insights into your project, the COCOMO II model does not stand out from other estimation methods. Estimation is an imprecise process, and as McConnell states in *Code Complete* [5]:

> Just as an architect can't estimate how much a 'pretty big' house will cost, you can't reliably estimate a 'pretty big' software project. It's unreasonable for anyone to expect you to be able to estimate the amount of work required to build something when "something" has not yet been defined.

Thus, other methods of estimation could produce results than are just as reasonable as COCOMO II. COCOMO II was meant for aerospace applications, not HPC. It does poorly when modeling HPC projects, so it would not work well on the EHT project. It relies on parameters that may be inaccurate and hard to correctly characterize. It is not in widespread use in the software development industry, which shows that it is ineffective. Although in theory the model can handle different types of software well, in practice it fails to produce notable results. Learning to apply the model is not worth the effort.

The alternatives such as SEER-SEM, SLIM-Estimate, TruePlanning, or even informal estimates could produce as similar results with less effort. The alternative parametric models all produced comparable performance as COCOMO II [10]. But COCOMO II is almost 20 years old and hard to use, while these other models are newer and easier to use. They are backed by companies who can assist in the estimation process. The last option, informal estimates based on developer experience, is even easier to use since it can be made as complex or as simple as desired. So these models would provide a better overall experience and be easier to use on the EHT project.

6 Conclusion

In this debate essay I examined the various arguments for and against the use of COCOMO II in estimating the cost for developing the HPC software used in the EHT. On its own the COCOMO II model appears to be reasonable, but when considering alternative models it begins to lose its appeal. My final recommendation is that COCOMO II should not be used on the EHT project. Other parametric models or even informal estimates based on developer experience would be a better choice.

References

[1] Chris Abts, Bradford Clark, Sunita Devnani-Chulani, Ellis Horowitz, Raymond J. Madachy, Donald J. Reifer, Richard W. Selby, and Bert Steece. Cocomo ii model definition manual. 2000.

[2] Barry W. Boehm, Clark, Horowitz, Brown, Reifer, Chulani, Ray Madachy, and Bert Steece. *Software Cost Estimation with Cocomo II with Cdrom*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2000.

[3] Katherine L. Bouman, Michael D. Johnson, Daniel Zoran, Vincent L. Fish, Sheperd S. Doeleman, and William T. Freeman. Computational imaging for vlbi image reconstruction. 2015.

[4] The Event Horizon Telescope Collaboration. First m87 event horizon telescope results. ii. array and instrumentation. *The Astrophysical Journal Letters*, 875(1), 2019.

[5] Steve McConnell. *Code Complete, Second Edition*. Microsoft Press, Redmond, WA, USA, 2004.

[6] Julian Miller, Sandra Wienke, Michael Schlottke Lakemper, Matthias Meinke, and Matthias S. Müller. Applicability of the software cost model COCOMO II to HPC projects. *International Journal of Computational Science and Engineering*, 17(3):283, 2018.

[7] P. Musilek, W. Pedrycz, Nan Sun, and G. Succi. On the sensitivity of COCOMO II software cost estimation model. In *Proceedings Eighth IEEE Symposium on Software Metrics*. IEEE Comput. Soc, 2002.

[8] P.C. Pendharkar, G.H. Subramanian, and J.A. Rodger. A probabilistic model for predicting software development effort. *IEEE Transactions on Software Engineering*, 31(7):615–624, July 2005.

[9] Ian Sommerville. *Software Engineering*. Pearson, 10th edition, 2015.

[10] Derya Toka and Oktay Turetken. Accuracy of contemporary parametric software estimation models: A comparative analysis. In *2013 39th Euromicro Conference on Software Engineering and Advanced Applications*. IEEE, September 2013.