INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY

RESEARCH ARTICLE

Use of Fibonacci Sequence in Project Estimation

Chandrashekhar Chauhan[1], **Mayank Dakoliya**[1]*, **R K Sharma**[2]

1 Institute of Engineering and Technology, Indore, India
2 Jawaharlal Nehru Smriti Govt. PG College, Shujalpur

OPEN ACCESS

Received: 21-06-2023
Accepted: 25-07-2023
Published: 08-09-2023

Citation: Chauhan C, Dakoliya M, Sharma RK (2023) Use of Fibonacci Sequence in Project Estimation. Indian Journal of Science and Technology 16(33): 2649-2652. https://doi.org/10.17485/IJST/v16i33.1534

* **Corresponding author**.

mayank.dakoliya@gmail.com

Funding: None

Competing Interests: None

Published By Indian Society for Education and Environment (iSee)

ISSN
Print: 0974-6846
Electronic: 0974-5645

Abstract

Background: Estimation and planning are crucial for successful software development projects, ensuring project readiness and preventing potential issues. Planning poker, an effective estimation method for agile teams, combines professional judgment, analogy, and disaggregation for fast, accurate estimates. The estimation technique Planning Poker is frequently employed in agile software development. When submitting estimates, Planning Poker only uses specific numbers on its cards, such as 0, 1, 2, 3, 5, 8, 13, 20, 40, and 100. This is inaccurately referred to in this work as a Fibonacci scale. Despite the frequent use of the Fibonacci scale in agile estimation, it is unknown how it influences the estimation process. **Objectives:** We conducted a study based on a software provider who estimates projects using a variety of estimation methods. **Method:** We used the planning poker technique for estimation. **Findings:** Using a Fibonacci scale as opposed to the conventional linear scale resulted in a median decrease in effort estimates. As the developer's proficiency increased, the disparity between the effort estimates shrank. **Novelty:** For the purpose of estimating the project of a software development company that is presently in operation, we have utilized the currently available estimation method.

Keywords: Planning Poker; Effort Estimation; Fibonacci Sequence; Story Point; Fibonacci Scale

1 Introduction

Agile software development is a magnificent term that refers to a variety of frameworks and practices founded on the Agile Manifesto's stated values and principles. Agile software development is one of the most prevalent and fashionable methodologies [(1)] used in the Information and Communications Technology (ICT) industry, particularly in software industries where estimation is the determining factor.

Software engineering is an important field that concentrates on the development of software systems by establishing processes and practices that aim to ensure that projects are developed within the constraints of scope, cost, time, and quality. The waterfall model, which is a linear, sequential model that begins with analysis and is followed by design, implementation, testing, and deployment, was the dominant method for developing projects for a long time [(2)]. Agile development spawned numerous methodologies, including Scrum [(3)]. Scrum is an iterative project management

methodology that emphasizes the rapid and early delivery of functional software and the ability to adapt to changing requirements as the system is developed. In Scrum, the requirements for a project are articulated as a collection of user stories that are organized in a product backlog. At any particular time, the backlog comprises all unimplemented user stories and, consequently, the remaining work. A user story depicts a feature or requirement from the end-user or customer's perspective. It is a straightforward, concise, and non-technical description that conveys the essence of the user's desired outcome. The following format is utilized to briefly describe user stories: "As a [user type], I desire [objective] so that [reason or advantage]" Typically, this sentence is followed by additional details and acceptance criteria.

Expert estimation techniques such as Delphi's method, the wideband Delphi technique, Work Breakdown Structure (WBS), rule-based systems, and the top-down and bottom-up approaches [4] dominated traditional effort estimation. Over the years, these models have provided researchers with valuable datasets for constructing various approaches based on machine learning techniques to generate more accurate estimates. The algorithmic techniques adhere to mathematical equations, whereas the analogy-based or machine learning techniques are based on a dependable database and can be readily implemented in the early phases of a project [5]. Fernandez-Diego et al. [6]

The empirically founded function-related classifications of Artificial Intelligence (AI) in software engineering refer to the technologies of AI applied to the software engineering process [7], such as language processing, knowledge management, machine learning, and machine vision. A lack of data is one of the reasons why AI has not been extensively adopted in agile development. In some instances, agile project datasets are private, and only handfuls are available in online repositories [8]. Existing effort estimation methods can be categorized as expert judgment, algorithmic, machine learning, and statistical [9], and it is well known that expert judgment has been extensively used since the 1980s. In addition, other metrics, such as function points and use case points, have been applied to estimation, and studies on machine learning [10,11] focus on non-agile models.

2 Methodology

Estimating the software work is a crucial task for software project management participants. The difficulty of estimating effort is exacerbated by the constant evolution of software development. In the last three decades, numerous methodologies for estimating software development costs have been developed. There are numerous cost estimation methodologies, algorithmic models, non-algorithmic models, and machine learning techniques from which to choose.

We conducted a study based on a software company that employs a variety of project estimation techniques. We estimated using the planning poker method.

We have chosen two teams of six members each and told them to estimate their efforts on any scale. Then we use the Fibonacci scale to compare their estimation with the scale they used.

The Planning Poker method operates as follows:

Step1. Preparation: The product owner or scrum master creates a list of user stories or tasks that need to be estimated prior to the estimating session. The team should have a firm understanding of each user's narrative. Estimation session,

Step2: For the estimating session, the team meets up either physically or virtually, utilizing Planning Poker-specific software or tools. A deck of Planning Poker cards or access to a digital tool is given to each team member.

Step3. Fibonacci sequence: The most typical deck of cards used in planning poker has the numbers 0 through 21 (the Fibonacci numbers are 0, 1, 2, 3, 5, 8, 13, and so forth). Larger numbers denote greater complexity, and these values represent the relative effort or difficulty of the user stories.

Step4: Establishing Consensus The product owner or scrum master reads a user story, and the team clarifies any ambiguities or queries to reach consensus. Members of the team secretly choose a card to symbolize their estimate at this point.

Step5: Discussion and revelation: After everyone has selected a card, the team members all simultaneously share their estimations. If there is a broad range of estimations, the team has a conversation to comprehend various viewpoints, define assumptions, and resolve any issues.

Step6. Repeat Estimation: The team iterates the estimating process for the same user narrative until a decision is made in. With this iterative process, conversations and modifications are possible based on common knowledge and understanding.

Step7: Final calculation: The group documents the user story's agreed-upon estimate after reaching a consensus. The Planning Poker card selected to symbolize this estimate is usually used.

For further user tales, repeat: The team then repeats stages 4 through 7 while moving on to the next user narrative.

3 Result and Discussion

For this study, we have chosen two software development teams of six members each. Whose task it is to implement a feature list for data analysis software, the features that they have to develop are:

1. List of feature - Base price, distributions, total value spends etc
2. Data Source - which feature is taken from which source?
3. Data granularity - what is the level of granularity i.e. Brand level, Brand Retailer level
4. Time granularity - Data is weekly/monthly or periodic
5. Availability time period - How many days / month of data is available
6. Aggregation method- How to aggregate the data at higher level e.g. price will be averaged spend will be summed

We have used the planning poker method to estimate the effort required for this endeavor. There are currently six features to develop, or six user stories to execute.

All participants were given a real-world requirement specification following an introduction to the Planning Poker estimation method. In total, each developer and team estimated six requirements. The estimated effort for a requirement should include development and unit testing. Before beginning the Planning Poker (group-based) estimation, each participant independently estimated the amount of work necessary to develop software that meets the six requirements. These individual estimates were not constrained by any scale format instructions; participants were simply instructed to jot down the number of work hours required to complete the tasks. This session suggests that there was no difference in scale utilization for the initial, individual estimate, only for the teams' revisions to their estimates during the Planning Poker sessions. Each team consisted of six members with experience with the same platform who utilized the Planning Poker estimation technique for their group-based estimation work. If no consensus was reached after four rounds of Planning Poker, the team's ultimate effort estimate for a requirement was the mean of the estimates. The requirement estimation cards had numbers based on a Fibonacci or linear scale.

Fibonacci: 0, 1, 2, 3, 5, 8, 13, 21, 34, 55,...

Linear: 1, 2, 3,...

One team estimated assignments (user stories) 1–6 using Fibonacci scale cards, while the other used linear scale cards. Evidently, those utilizing the linear scale did not have access to the values 0 and 1. Since there were no estimates that were meaningful with zero work hours, there were none.

Table 1. Fibonacci Scale

Story point	Effort required	Time required	Complexity	Risk/ uncertainty
0	No effort	A few minutes	None	None
1	Minimum effort	a few minutes	Little complexity	None
2	Minimum effort	A few hours	Little complexity	None
3	Mild effort	A day	Low complexity	Low
5	Moderate effort	A few days	medium complexity	Moderate
8	Sever effort	A week	medium complexity	Moderate
13	Maximum effort	A month	High complexity	High
21/34	Can not be completed in expected time frame	More than a month	——–	———–

Table 2. Estimation of developers (using Fibonacci)

Round		D1	D2	D3	D4	D5	D6
1		3	3	2	3	2	3
2	Story point	3	3	3	2	3	3
3		3	3	3	3	3	3

Where D1,D2, …..,D6 are developers

After 3 planning poker round the team agrees to 3story point which means the task can be completed in a day.

Table 3. Estimation of developers (using other scale)

Round		D1	D2	D3	D4	D5	D6
1		3	4	4	5	5	6
2	Story point	3	4	4	6	5	6
3		4	5	4	6	6	6

Where D1, D2, … D6 are developers

After 3 rounds of using any liner sequence another team didn't agree on same result so the average of this i.e 5.16 ≈ 5 days which is lager in comparison of Fibonacci scale.

3.1 Limitations

The Fibonacci sequence is often used in agile estimation contexts, but it can lead to errors such as converting the sequence into hours, a lack of shared understanding, and misunderstandings among team members. Software engineers may dislike estimating due to its abstract nature, and team members may advocate for task division into technical aspects. Different perceptions of transfer duties can also affect estimates. The estimation procedure uses a linear scale format, but it is unlikely to be used without negative side effects. The impact of non-linear scales on larger tasks and initiatives is uncertain, and studies on actual effort are needed to provide conclusive evidence.

Further research is needed to understand the effect of non-linear scales on effort estimates. This research is a preliminary step towards a more complete understanding of the Fibonacci sequence and its implications for enhanced estimation accuracy in agile and other contexts.

4 Conclusion

When designing a software project, there are three fundamental elements to consider: resources, scope, and delivery date. To ensure quality development, projects must be completed on schedule, administrative costs must remain within budget, and customer expectations must be satisfied.

Our Research shows scale choice affects effort estimates, especially with substantial uncertainty. Software developers should be cautious when using non-linear scales like Fibonacci, which may expedite estimation but may bias towards insufficient effort values.

References

1) Govil N, Sharma A. Validation of agile methodology as ideal software development process using Fuzzy-TOPSIS method. *Advances in Engineering Software*. 2022;168:103125. Available from: https://doi.org/10.1016/j.advengsoft.2022.103125.
2) Wysocki RK. Effective Project Management: Traditional, Agile, Hybrid, Extreme. Hoboken, NJ, USA. Wiley. 2019. Available from: https://asalraac22.files.wordpress.com/2018/04/effective-project-management-traditional-agile-extreme-7th-edition.pdf.
3) Sommerville I. Software Engineering. 2019. Available from: https://engineering.futureuniversity.com/BOOKS%20FOR%20IT/Software-Engineering-9th-Edition-by-Ian-Sommerville.pdf.
4) Rashid J, Nisar MW, Mahmood T, Rehman A, Syed YA. A study of software development cost estimation techniques and models. *Mehran University Research Journal of Engineering and Technology*. 2020;39:413. Available from: https://doi.org/10.22581/muet1982.2002.18.
5) Hoc HT, Van Hai V, Nhung HLTK. A Review of the Regression Models Applicable to Software Project Effort Estimation. *Computational Statistics and Mathematical Modeling Methods in Intelligent Systems*. 2019;2:399–407. Available from: https://doi.org/10.1007/978-3-030-31362-3_39.
6) Fernandez-Diego M, Mendez ER, Gonzalez-Ladron-De-Guevara F, Abrahao S, Insfran E. An Update on Effort Estimation in Agile Software Development: A Systematic Literature Review. *IEEE Access*. 2020;8:166768–166800. Available from: https://doi.org/10.1109/ACCESS.2020.3021664.
7) Barenkamp M, Rebstadt J, Thomas O. Applications of AI in classical software engineering. *AI Perspectives*. 2020;2(1). Available from: https://doi.org/10.1186/s42467-020-00005-4.
8) Dave CV. Estimation Approaches of Machine Learning in Scrum Projects: A Review. *International Journal for Research in Applied Science and Engineering Technology*;9(11):1110–1118. Available from: https://doi.org/10.22214/ijraset.2021.38977.
9) Sudarmaningtyas P, Mohamed R. A Review Article on Software Effort Estimation in Agile Methodology. *Pertanika Journal of Science and Technology*. 2021;29(2). Available from: https://doi.org/10.47836/pjst.29.2.08.
10) Kumar PS, Behera H, Nayak AKKJ, Naik B. Advancement from neural networks to deep learning in software effort estimation: Perspective of two decades. *Computer Science Review*. 2020;38. Available from: https://doi.org/10.1016/j.cosrev.2020.100288.
11) Mahmood Y, Kama N, Azmi A. A systematic review of studies on use case points and expert-based estimation of software development effort. *Journal of Software: Evolution and Process*. 2020;32(7). Available from: https://doi.org/10.1002/smr.2245.