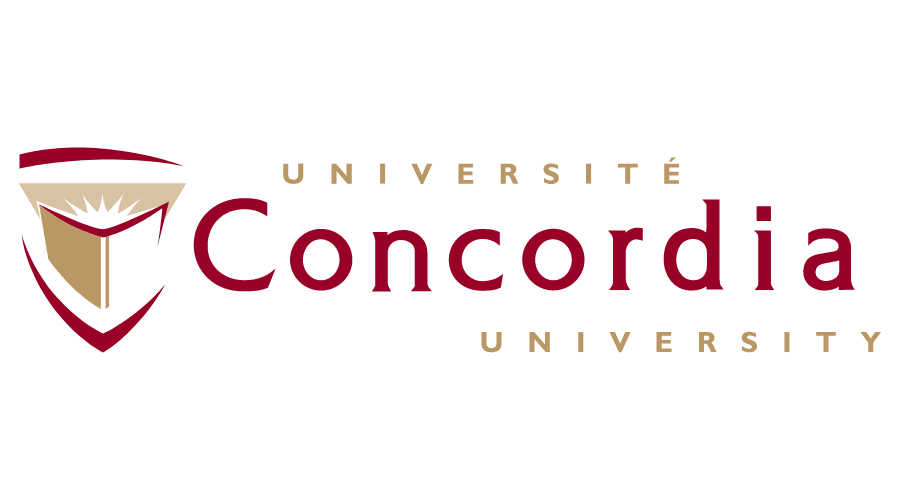
****

Montreal QC H3G 1M8, Canada   
[www.concordia.ca](http://www.concordia.ca)

**Masters of Engineering in Software Engineering**

**Subject: Software Project Management (SOEN 6841)**

**Title**

**Value Results, Not Just Effort**

Author

Som Mosalagi (Student ID: 40238546)

Github Link: <https://github.com/som-50111/SOEN6841_SPM_TAS>

Professor

Prof. Pankaj Kamthan

**Table of Contents**

1. Abstract …………………..…………………………………………. 3
2. Introduction …………………..…………………………………………. 3
3. Motivation …………………..…………………………………………. 3
4. Problem Statements …………………..…………………………………………. 4

3. Background Material …………………..…………………………………………. 4

1. Potential Questions …………………..…………………………………………. 4
2. Scenario …………………..…………………………………………. 6

4. Methods & Methodology …………………..…………………………………………. 6

5. Results Obtained …………………..…………………………………………. 7

1. Learnings …………………..…………………………………………. 7

6. Critical Thinking …………………..…………………………………………. 8

7. Conclusion & Future Work …………………..…………………………………………. 8

8. References …………………..…………………………………………. 9

9. Apendix …………………..…………………………………………. 10

**Table of Figures**

Fig 1. Graph showing Planned Effort vs. Actual Effort with Development Time …………….…… 5

1. **Abstract**

Although creating software requires a lot of work, judging success only by the quantity of work done might have unfavourable effects. The significance of appreciating software development results over just the quantity of code created is emphasized in this essay. A mistaken belief in extensibility frequently results in the introduction of excessive, out-of-scope code in software projects. The paper makes the case that extensibility should be handled carefully because writing too much code can cause delays in projects and a host of other problems.

The topic can be compared to gardening, where trees can be harmed by overwatering. Programmers, on the other hand, are more like trees in that they do best when given enough time and clearly defined jobs [9]. Working too many hours can result in complicated code and declining returns. The piece highlights how a results-oriented perspective, as opposed to a time-centric one, can result in more productivity and better outcomes by sharing experiences from various management techniques.

**Keywords**: work culture, collaboration, managerial emphasis, results-driven environment, software development teams, work-life balance, results-oriented management, burnout, employee well-being, job retention.

1. **Introduction**

An essential idea in software development is discussed in the article "Value Results, not just efforts"—emphasizing the caliber of results above the amount of work put in [9]. The introduction explores the widespread inclination for people to brag about how much code they write for a project, which makes the reader wonder if so much code is really necessary. By comparing overwatering a tree to other potential negative outcomes, we can draw attention to the potential consequences of excessive effort [3]. This introduction lays the groundwork for a more thorough explanation of the negative impacts of excessive, outside-the-scope code and the unfavorable outcomes of a management style that just encourages longer workdays without guaranteeing concrete, useful outcomes.

**2. a. Motivation**

The motivation for the title "Value Results, not just Efforts" stems from a desire to challenge prevalent norms in the software development industry and advocate for a paradigm shift in how success is measured [1]. Several key motivations drive the topic’s intent [2]: Challenge Conventional Boasting, Highlight Counterproductive Practices, Analogies for Enhanced Understanding, Address Management Issues, Share Personal Experience, Advocate for a Results-Oriented Approach, Improve Work Conditions for Programmers, Encourage Reporting of Progress

Overall, the motivation is to provoke thoughtful consideration of prevalent practices, challenge assumptions, and inspire a positive change in the mindset and management approach within the software development community.

**2. b. Problem Statement**

There is a common problem in modern software development settings where the focus is more on effort than on the critical assessment of observable outcomes, as measured by metrics like lines of code produced and hours spent [2]. The addition of superfluous, out-of-scope code is a manifestation of this emphasis on mere quantity above essential and useful output, which is made worse by a management style that encourages lengthy work hours. Project delays, worse code quality, and a possible drop in the general productivity and well-being of software engineers are all effects of this issue. Recognizing that excessive effort does not always convert into significant achievements, addressing this issue requires a move towards a results-oriented strategy.

**2. c. Objective**  
  
The objective of the topic is to:

1. Challenge Conventional Metrics
2. Highlight the Pitfalls of Excessive Code
3. Address Managerial Practices
4. Draw Analogies for Understanding
5. Promote Effective Work Time Management
6. Share Personal Experience and Real-world Analogies
7. Highlight the Impact of Managerial Focus
8. Encourage a Results-Oriented Culture
9. Improve Team Focus and Productivity
10. Facilitate a Shift in Team Focus

The overall goal is to question accepted wisdom, expose ineffective behaviors, and push for a change in team dynamics and software development management towards a results-driven methodology [2].

**3. Background Material**

**3. a. Potential questions**

1. How does a managerial emphasis on hours worked, rather than results produced, impact the efficiency and effectiveness of software development projects?

The research might find that a strong emphasis on hours worked tends to lead to burnout, decreased job satisfaction, and potentially a negative impact on the quality and efficiency of software development projects.

1. In what ways does a focus on long working hours contribute to the creation of unnecessary or out-of-scope code in software development, and how does this affect project timelines?

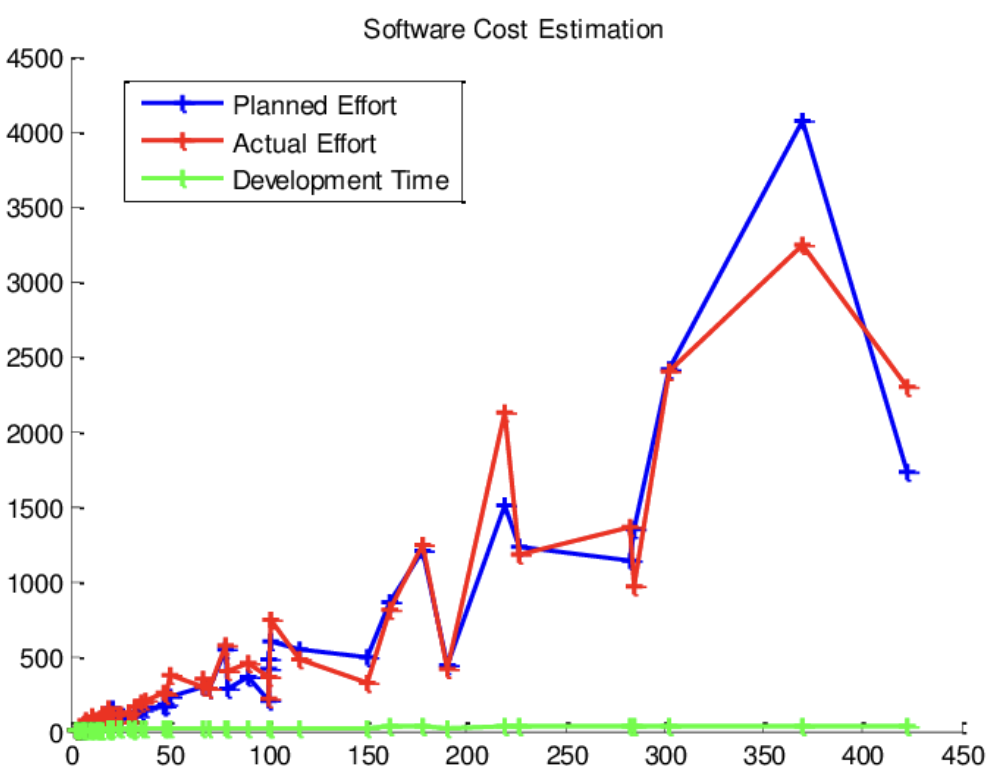
The study could reveal that a managerial push for extra effort without a focus on results may contribute to the inclusion of unnecessary or out-of-scope code, leading to project delays and increased complexity.

1. What are the effects of different task allocation strategies (small, adequate amounts of time vs. larger task chunks) on the productivity and code quality of software developers?

Research might indicate that smaller, well-defined tasks with adequate time allocation lead to higher productivity and better code quality compared to larger task chunks and extended working hours.

1. How does a results-oriented management approach, as opposed to one focused on tracking hours, influence the motivation and job satisfaction of software developers?

Results may show that a results-oriented management approach positively influences motivation and job satisfaction among software developers, fostering a healthier work environment.



# Graph showing Planned Effort vs. Actual Effort with Development Time. [8]

Effort adds value to the products of effort, but effort itself also has value. Effort’s value can not only be accessed concurrently with or immediately following effort exertion but also in anticipation of such expenditure, suggesting that we already have an intuitive understanding of effort’s potential positive value [6].

**3. b. Scenario**

Companies typically use clear fonts and bright pictures in their ads, Web sites, and product-package designs; place their products on easy-to-reach shelves; and emphasize ease-of-usage to make their products appear desirable to consumers. However, we suggest that customers focused on “incentive” value (getting the best product) may instead see products associated with noninstrumental (pointless) effort as more desirable. We suggest that because effort is usually required to get the best outcomes, people looking for the best outcomes also mistakenly presume effort must imply the best possible outcome [7].

**4. Methods & Methodology**

It’s up to you to let your team know that even though they straddle both economies, as team members in an entrepreneurial environment, it’s only by keeping an eye on results that they or the company will be successful [1]. There are some key strategies to help your team get on board with your thinking. Remember, for you this is instinctive and intuitive. For your team, it isn’t until they learn your approach.

1. **Get aligned on what a great result looks like.**

"What does this look like when it's done and done well?" is one of the questions to ensure that people ask and to address. Be clear about who you are doing it with, when, how much, and how well [1]. Your team can become in sync with your vision and you won't be culpable of "drive-by delegations" if you clearly define success.

1. **Be coachable.**

It's important to communicate both ways, and I constantly urge team members to inform their entrepreneurs about any roadblocks [1]. Give your team authorization to inform you if they anticipate a longer turnaround time than two weeks for a task that you need completed in two days. Your team will be able to discover solutions and achieve outcomes more quickly if they are honest about the difficulties they are facing. It allows you to participate by offering quick fixes, different strategies, and shortcuts.

1. **Collaborate.**

Your project will turn out far better than you would have imagined if you worked alone by utilizing the skills on your team. Team members are even more productive when they have a personal stake in the tasks they are completing and are aware that the outcome matters [1]. I produce The Team Success Podcast series with a team, and we cooperate and work together at every turn. This is the reason it's so fun.

1. **Focus on constant improvement.**

Encouraging your staff to challenge themselves to perform their jobs in a faster, easier, cheaper, and better way is one method to keep people focused on results all the time. Everybody has access to technology that makes use of our abilities [1]. Your team will do more tasks in less time and with less effort if they concentrate on being even more resourceful.

1. **Results Obtained**

While putting in effort is an important factor in achieving success, there are various other elements that contribute to the outcomes [4]. These may include external factors, unforeseen circumstances, the complexity of the task, and sometimes just sheer luck. Efforts increase the likelihood of success, but they don't eliminate the possibility of failure. Success often involves a combination of hard work, strategy, timing, and sometimes factors beyond one's control. Additionally, the definition of "required results" may vary based on individual expectations, goals, and circumstances [4]. What is considered a success for one person might be viewed differently by another.

Evaluating the satisfaction of results from an effort involves considering various factors and can be subjective. Here are some key aspects to consider: Goal Achievement, Quality of Output, Learning and Growth, Efficiency, Feedback and Recognition, Adaptability, Long-Term Impact, and Personal Satisfaction.

It's important to note that satisfaction with results can vary from person to person, and external validation is just one aspect of evaluation [4]. Sometimes, despite external factors, if you feel a personal sense of accomplishment and growth, that can be a significant indicator of the success of your efforts.

**5. a. Learnings**

**Is it possible to learn from failures even if we don't achieve our goals?**

Absolutely, learning from failures is a crucial aspect of personal and professional development. While not achieving specific goals may be disappointing, failures offer valuable opportunities for growth, improvement, and gaining insights. Here are some ways you can learn from failure:

1. **Identify Mistakes and Weaknesses:** Analyze what went wrong and identify any mistakes or weaknesses in your approach. Understanding the root causes of failure helps you address those issues in future endeavors.

3. **Feedback and Evaluation:** Seek feedback from others, especially if they were involved in the project or task. Honest feedback can provide valuable insights and perspectives that you might not have considered.

4. **Adjust Strategies and Plans:** Use the lessons learned to adjust your strategies and plans for future endeavors. Failure can be a powerful teacher, helping you refine your approach and decision-making.

5. **Skill Development:** Consider what skills or knowledge gaps contributed to the failure. Use this information to prioritize skill development and learning opportunities.

7. **Refine Goals and Expectations:** Adjusting your goals based on what you've learned can set more realistic expectations for future endeavors.

8. **Persistence and Determination:** Developing persistence and determination enables you to keep pushing forward despite failures.

Learning from failures is an ongoing process that contributes to personal and professional growth. Embracing a positive attitude towards failure, viewing it as a source of valuable information, and using it as a catalyst for improvement are essential aspects of turning setbacks into opportunities for success.

1. **Critical Thinking**

Critical thinking in these areas will enable a more nuanced understanding of the dynamics between efforts, results, and effective management in software development [10].

* Interrogating Perceived Productivity: Challenge the conventional notion that longer working hours equate to higher productivity. Explore instances where extended work hours may lead to diminishing returns and how this relates to the software development process.
* Analyzing Extensibility in Software Development: Delve into the concept of extensibility in software projects. Question how it is commonly perceived, its benefits, and potential pitfalls.
* Comparative Analysis of Management Styles: Compare and contrast the outcomes of different management styles, particularly those that focus on hours worked versus results achieved. Assess the impact on team morale, creativity, and the overall success of software development projects.
* Applying Analogies to Software Development: Evaluate the effectiveness of the analogy between watering plants and managing software development. Consider how principles of balance, moderation, and targeted efforts in gardening translate to efficient code development and project management.
* Examining Employee Well-being: Consider the well-being of software developers in environments that prioritize results over extended work hours. Evaluate how such an approach affects job satisfaction, work-life balance, and the long-term commitment of team members.

1. **Conclusion**

In conclusion, the analogy drawn between nurturing a lawn and managing software development projects serves as a compelling reminder that quantity does not always equate to quality. The essay highlights the detrimental impact of excessive code, often added under the guise of extensibility, and the consequences of valuing long hours without ensuring tangible results. Through real-world examples, it becomes evident that a results-oriented approach is paramount in ensuring project success and efficiency.

Encouraging programmers to prioritize reporting progress over hours worked is a key takeaway. When team members understand that the emphasis is on achieving meaningful results rather than clocking hours, a shift towards a more productive and motivated workforce is inevitable.

**Future Work**

Exploring ways to strike a balance between extensibility and streamlined code development is another area ripe for future research. Techniques and best practices for optimizing codebases without sacrificing flexibility could be a valuable contribution to the software development community. Moreover, delving into the broader implications of a results-driven mindset on organizational culture and employee well-being would provide a holistic perspective.

In essence, the essay not only advocates for a change in mindset but also opens the door to a plethora of opportunities for further investigation into refining project management practices and cultivating a work environment that truly values results.

1. **References**
2. <https://resources.strategiccoach.com/the-multiplier-mindset-blog/focus-your-team-on-results-not-time-and-effort>
3. https://grantcardone.com/results-matter-more-to-the-successful-than-effort/#:~:text=Successful%20people%20don't%20value,the%20concept%20of%20being%20unreasonable
4. <https://www.linkedin.com/pulse/reward-results-effort-bob-ashenbrenner/>
5. <https://www.inc.com/josh-linkner/efforts-vs-results.html>
6. LNCS Homepage, <http://www.springer.com/lncs>, last accessed 2023/10/25
7. Inzlicht, Michael, Amitai Shenhav, and Christopher Y. Olivola. "The effort paradox: Effort is both costly and valued." Trends in cognitive sciences 22.4 (2018): 337-349.
8. Sara Kim, Aparna A. Labroo, From Inherent Value to Incentive Value: When and Why Pointless Effort Enhances Consumer Preference, Journal of Consumer Research, Volume 38, Issue 4, 1 December 2011, Pages 712–742, <https://doi.org/10.1086/660806>
9. <https://www.researchgate.net/figure/Graph-showing-Planned-Effort-vs-Actual-Effort-with-Development-Time_fig2_264042862>
10. chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/<https://users.encs.concordia.ca/~kamthan/courses/soen-6841/topics/113.pdf>
11. ChatGPT (https://chat.openai.com/c/d3fc9c16-ba01-4ca8-9bfe-391eca7b05a4)

**9. Appendix**

1. How does critical thinking matter?
2. How do we identify if critical thinking is necessary during putting efforts into a software development?
3. What is the future scope in a software development life cycle when we compare values with efforts?
4. What are the learnings from the topic “Value Results, not just Efforts”?
5. What are the chances of failure(s) even after putting right efforts?