[1]

Measuring Software Engineering

SOFTWARE ENGINEERING

Vince Casey | 18327140 | 27/11/20

[1] https://azure.microsoft.com/en-us/services/devops/

Table of Contents

Brief

To deliver a report that considers the ways in which the software engineering process can be measured and assessed in terms of measurable data, an overview of the computational platforms available to perform this work, the algorithmic approaches available, and the ethics concerns surrounding this kind of analytics.

Introduction

Why Measure Software Engineering?

It is in our nature as computer scientists to want to be as efficient, as optimal and as high quality possible in what we do. Knowing this it is only natural that measuring software engineering is held up as one of the most important aspects of the field. Through measurement we can learn how to save time and money as well as learning how to produce better products and software.

In this report, I will attempt to explain how to measure software engineering as well as delving into some of the ethical implications behind a lot of applications of the practice. I will also use some of my own experience using Azure DevOps and an Agile workflow at companies as a software engineering intern to give examples of

Measuring Data

First, we must consider:

What Is Data?

For the purposes of this report data is define as “information, especially facts or numbers, collected to be examined and considered and used to help decision-making.”[2] From this we can infer that whatever data we examine should be useful in regards to making decisions in whatever software engineering project is being worked upon. That brings us to our next question:

[2] https://dictionary.cambridge.org/dictionary/english/data

Who Are We Collecting Data From?

While we are obviously collecting data from software engineers, who those engineers are strongly influences both what we chose to measure and what we chose to do with that data. A software engineer could be a web developer at a major company or could be a student, working on an assignment for college. We could also be collecting data from a team or even a whole company or organization. The kinds of data we collect on how we chose to interpret it will always depend on these factors, so it is important that this is acknowledged throughout the paper.

What Data Can Be Measured?

- **Time Taken** – Usually measured in how many hours spent on a particular task or project.

- **Objectives/Tasks Completed** – On a particular project, every piece of work can be broken into basic tasks. For example, when using the Agile approach to software development tasks can be broken up into Themes, Initiatives, Epics and Stories with stories being the most basic task for a program. If needed, stories can also have Tasks or "children" (thus making them the parent) for sub tasks of the given story.

- **The Amount of Code Written** - Generally measured by the number of lines written by the developer. This can include or exclude lines of code that were removed from later versions of a project.

- **Cost** – The amount of money spent on a given project. Typical costs include any fees for services used by engineers and wages.

- **Quality of the Code Committed/Number of Commits** – The quality of code is something that cannot be measured in and of itself, however there are various methods that while not true measurements of such an undefined concept, can indicate whether code was of higher quality or not.

  For example, the number of commits to a particular task or story can be a very useful indicator of the initial quality as more commits under the same objective can indicate worse quality of code.

Another metric would be the amount of changes made or suggested during what is called a Pull Request. When working at my internship when we would finish a Task or User Story, we would submit a Pull Request. Another member of the team would then have to review this code and make comments if any changes needed to be made. All comments must be resolved before a pull request can be merged with the main repository.

What Data Should Be Measured?

Defining what data is worth measuring is determined by what information you make available by collecting and analyzing it. A useful concept when it comes to judging such data is the idea of Software Productivity.

Software productivity can be defined as the ratio between the functional value of software produced to the labor and expense of producing it.[3]

From this definition we see two metrics to judge our productivity from:

1. The functional value of software produced – The Objectives/Tasks Completed from above.
2. The efforts and expense required for development – Time Taken, The Amount of Code Written and Cost.

Time Taken

Time taken is the quintessential measurement of productivity. However, one must be very careful not to use it without the context of other measurements as doing so can lead to a misinformed extrapolation of what the data is telling you.

For example, one engineer may implement a vast array of new functionality to a project while another adds very little in a very short space of time. However, upon further inspection you could find that the second was spending a considerable amount of time on more difficult tasks whereas the first engineer was simply making several small changes that were all easily implementable.

3 http://www.umsl.edu/~sauterv/analysis/488_f02_papers/SoftwareProductivity.html

Other context outside of our already defined metrics is also important. For example, an engineer could be learning a new language or application or be working on other non-software engineering tasks to do with the project at a time.

Another important factor to bear in mind is often code of worse quality can be completed and submitted in much shorter timespans than better iterations. This is not a good idea as it will often require revisions in the future and perhaps lead to some misleading figures in terms of time spent on it.

Objectives/Tasks Completed

The number of objectives or tasks completed can be a very useful measurement however it can often require a lot of organization on the part of the engineer or team. Tasks and objectives must be planned out in advance and have stories, epics etc. created for them to organize the engineers work in advance. This can all be measured and kept track of using software which I will talk in more detail about in the “What can be used” section of this report.

The Amount of Code Written

Writing larger amounts of code will often indicate higher amounts of time and effort put into a project however something that is important to note is that they are not necessarily indicative of a higher amount of functional value that has been added to the software. It is also not indicative of the quality of code written and in fact often much more concise code will be more efficient for a program to execute. As such the amount of code written while easy to measure and compare to other engineer’s output, is not a hugely useful piece of information to extrapolate compared to some other measurements.

Cost

Cost is one of the most important factors in measuring software engineering in a business/company. There are two aspects of cost to consider when making decisions based off measured information. The first is of course the cost of production or how much money is spent adding a particular feature to a project or making the project. The second is how much revenue that project or feature will generate. When working on a project it can sometimes be worth it to make more expensive investments when seeking greater returns however in doing this you also risk more losses of income if that given feature is delayed, cancelled or does not sell as well as predicted.

Quality of the Code Committed/Number of Commits

The quality of code can be a very useful statistic to have for your project. Although hard to define, guidelines like the number of commits and comments on those commits can paint a very strong picture as to the ability of a given engineer. One useful use of this is to look at an engineer's quality statistics over time and see how they improve as they adapt and become accustomed to a given companies languages, applications and coding or commit message standards.

When managing a team, this information can be very useful in deciding who to assign tasks and stories to depending on what types of objectives workers have been strong at in the past. It can also inform decisions on who to give more important positions or even who to remove from a project team.

Measuring Data in Teams

When working in teams, there are various other important skills important for individuals on the team that will increase productivity of the team. Working with a team can present many challenges.

Task Assignment

One example of this is the assignment of different tasks and stories to individuals or groups in the team. It is important to ensure that the work assigned is both somewhat fair in terms of who must do what and how much every person must do. This can be calculated with a lot of the previous statistics such as time taken, objectives/tasks completed, and the amount of code written. However, it is also important to ensure the correct individuals are put on the appropriate tasks to ensure the work is done optimally and efficiently. This can be calculated by the quality of code as discussed prior.

Code, Comment and Commit Message Readability

It is very important, when working in a team, for others involved in the project either currently or those who will be in the future to be able to read and understand your code, comments and commit messages. This will make things much easier for them when making any changes or reviewing your code. Companies or groups will often use a coding standard as well as a commit message standard to ensure that there is consistency in how these things are written.

For example, during my internship we had 7 rules for how to write our commit messages which are outlined in the footnote below[4].

Catchup Meetings

One thing I found extraordinarily helpful over the course of my internship in terms of measuring progress in our work were the daily and weekly catchup meetings. These meetings allowed us to talk about our progress and mention any issues or queries we had and get help if needed. This both helped us be more efficient and helped us measure each other's progress which could inform our managers decisions on where steps next to take on the project we were working on.

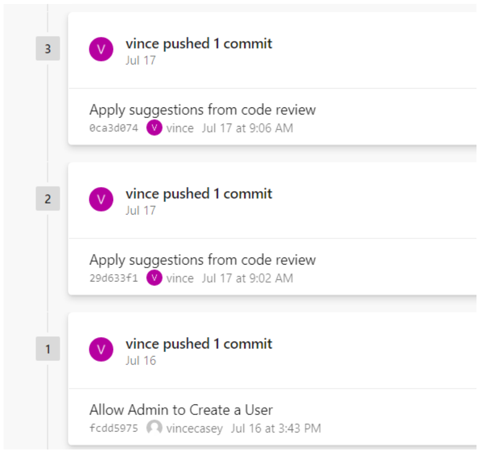
What Platforms Can We Use to Measure Data?

Now that we have established the reasons one would want to measure data about the software engineering process and what kinds of data is best to collect and use to make decisions, we can now discuss what platforms are useful for such data collection and analysis. When working at my internship we use Azure DevOps to monitor our progress and organize our work. It naturally organizes and compiles various information about our project and is superb for measuring the data we need.

When using a workspace such as Azure DevOps, engineers will first create or be assigned by a manager their tasks, stories, epics etc. Engineers can mark tasks as complete and submit those tasks to the branch for the Story or Task on the main repository. Then they can make a pull request when it is ready to be merged into the main branch.

Every commit, comment, completion of a task and more is all recorded by DevOps. All this information intricately maps out an engineer's workflow, process, and schedule. This provides an extraordinary amount of information to learn from alongside inherently having more context on what all this data means thanks to the sheer amount of information available.

[4] https://chris.beams.io/posts/git-commit/



3
vince pushed 1 commit
Jul 17

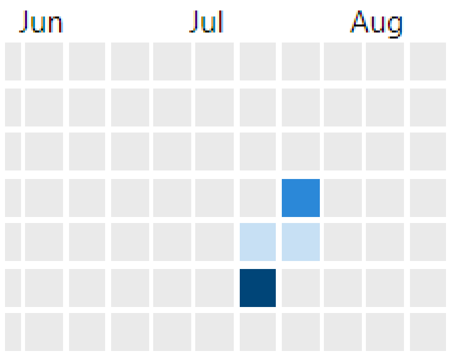
Apply suggestions from code review
0ca3d074 vince Jul 17 at 9:06 AM

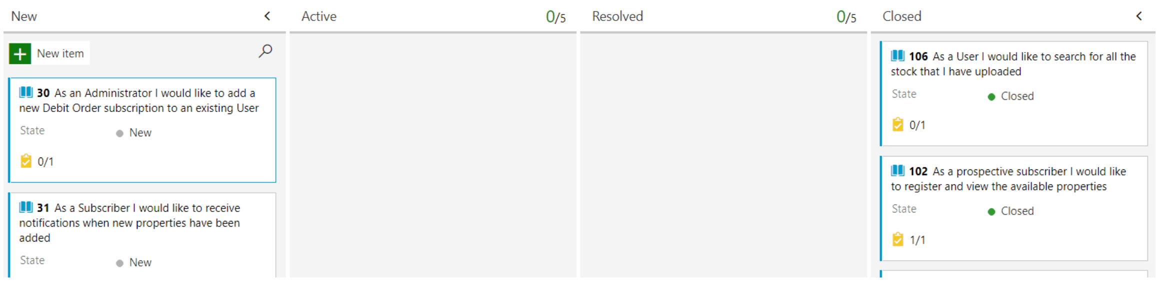
2
vince pushed 1 commit
Jul 17

Apply suggestions from code review
29d633f1 vince Jul 17 at 9:02 AM

1
vince pushed 1 commit
Jul 16

Allow Admin to Create a User
fcdd5975 vincecasey Jul 16 at 3:43 PM





New | Active 0/5 | Resolved 0/5 | Closed

New item

30 As an Administrator I would like to add a new Debit Order subscription to an existing User
State New
0/1

31 As a Subscriber I would like to receive notifications when new properties have been added
State New

106 As a User I would like to search for all the stock that I have uploaded
State Closed
0/1

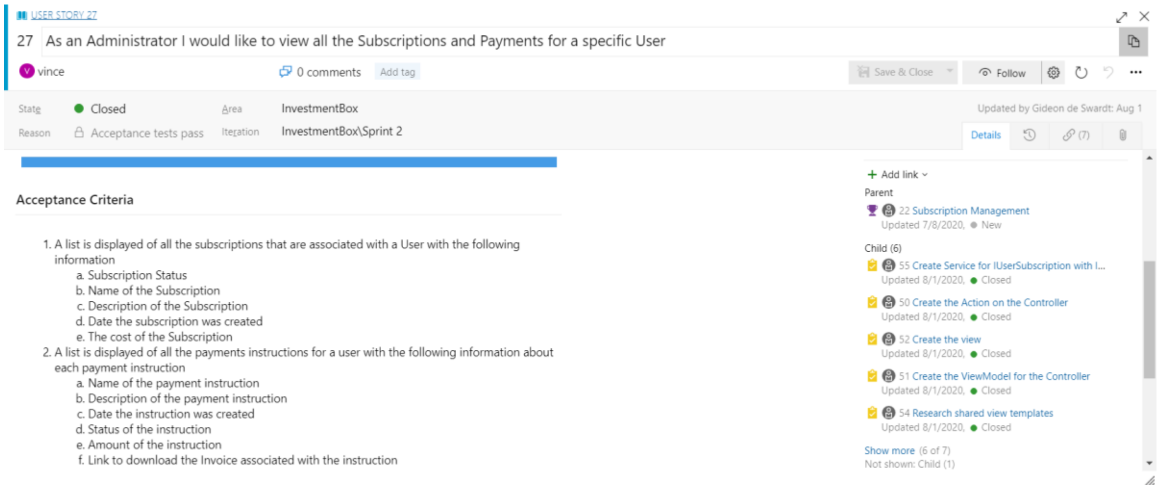
102 As a prospective subscriber I would like to register and view the available properties
State Closed
1/1

The above photos show some of the useful features Azure Dev Ops employs to enable users to measure their progress and data.

The first is a timeline of my commits I made. These were all done on a new branch I made for the feature. You can see that I have suggestions from a "code review" than I amended in my code. Thanks to Azure DevOps architecture, whenever I submit a pull request, members of my team can make comments and each comment is marked as a change needed to be resolved. I can only submit my code for another review once I have resolved every change. Once my code no longer has any suggested changes and has been reviewed by another member of the team, both I and that member can approve the code and it will be merged into the main branch.

The second picture is a graph of work I did on a particular project in a particular period. The amount of work I did is signified by the shade of blue used in the graph.

The third picture is what a board looks like on the platform. Stories are displayed here on the board and divided by their status which can be either New, Active, Resolved or Closed. This is brilliantly useful for keeping track of progress and how close each individual feature is to completion.


USER STORY 27

27 As an Administrator I would like to view all the Subscriptions and Payments for a specific User

vince · 0 comments · Add tag

State: Closed · Area: InvestmentBox

Reason: Acceptance tests pass · Iteration: InvestmentBox\Sprint 2

Updated by Gideon de Swardt: Aug 1

Acceptance Criteria

1. A list is displayed of all the subscriptions that are associated with a User with the following information
   a. Subscription Status
   b. Name of the Subscription
   c. Description of the Subscription
   d. Date the subscription was created
   e. The cost of the Subscription
2. A list is displayed of all the payments instructions for a user with the following information about each payment instruction
   a. Name of the payment instruction
   b. Description of the payment instruction
   c. Date the instruction was created
   d. Status of the instruction
   e. Amount of the instruction
   f. Link to download the Invoice associated with the instruction

Parent
- 22 Subscription Management — Updated 7/8/2020, New

Child (6)
- 55 Create Service for IUserSubscription with I... — Updated 8/1/2020, Closed
- 50 Create the Action on the Controller — Updated 8/1/2020, Closed
- 52 Create the view — Updated 8/1/2020, Closed
- 51 Create the ViewModel for the Controller — Updated 8/1/2020, Closed
- 54 Research shared view templates — Updated 8/1/2020, Closed

Show more (6 of 7)
Not shown: Child (1)

This is a closer look at an individual User Story on the board. On this page I can view the various children (tasks) needed to complete the story, read over my plan for the implantation and its acceptance criteria, check the status of the story and more.

Another very useful platform used over the course of my internship was Slack. Slack is a fantastic resource that enables groups to organize various text channels to coordinate communication within a team or company. As discussed previously, communication is pivotal in working on a project with a team and as such, we used Slack daily. Typical examples of my use of the platform included tagging members of my team using the "@channel" functionality in order to ask them to review my pull request, asking members of my team about any queries I may have, Informing my team about any breaks I may be taking and much more.

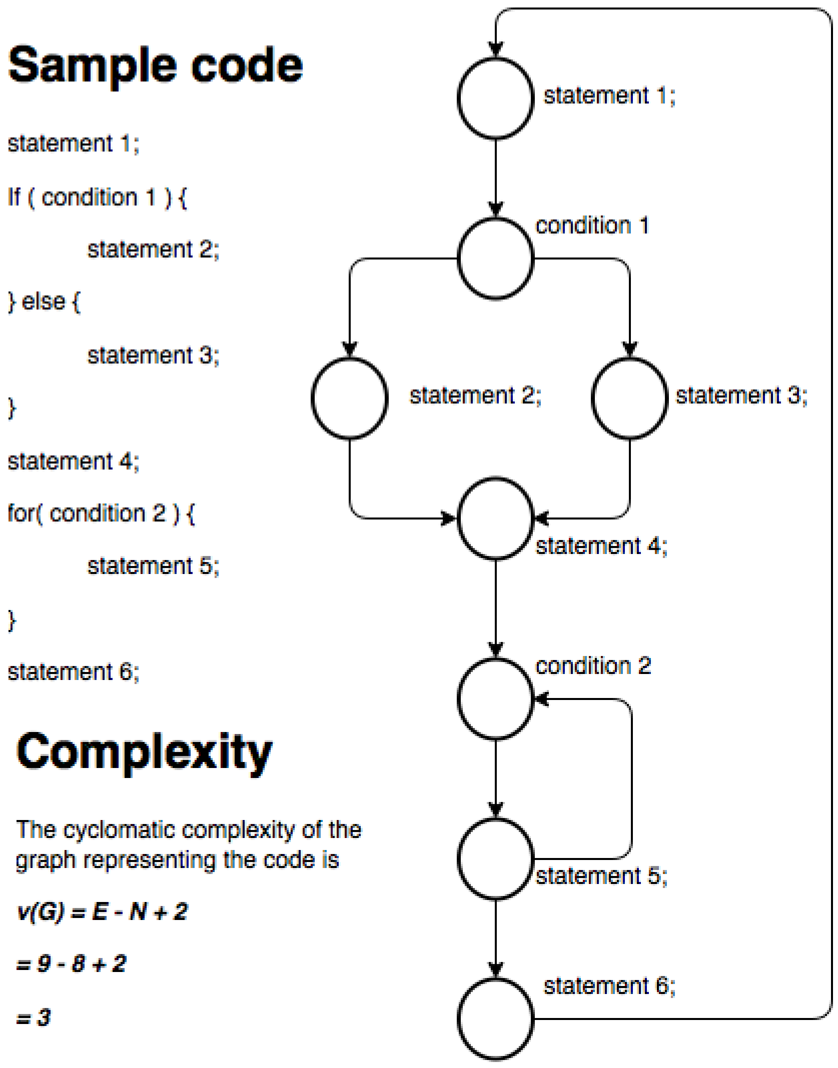
What Algorithms Can Be Used?

There are also many algorithmic approaches to measuring software engineering.

Cyclomatic Complexity

The amount of code in a project is not indicative of how complex that code is. As such we must employ other methods to measure the complexity (and by extension the efficiency) of the code in each project.

Cyclomatic Complexity can be calculated with the formula ***v(G)*** = ***E*** - ***N*** + ***P*** where v(G) is the cyclomatic number, G is the graph or program in question, E is the number of edges on the graph, N is the number of vertices and P is the number of connected components. The following is an example taken from a program:



Sample code

```
statement 1;
If ( condition 1 ) {
        statement 2;
} else {
        statement 3;
}
statement 4;
for( condition 2 ) {
        statement 5;
}
statement 6;
```

Complexity

The cyclomatic complexity of the graph representing the code is

v(G) = E - N + 2

= 9 - 8 + 2

= 3

[5]

[5] https://www.softwareyoga.com/cyclomatic-complexity/

The Ethics of Measuring Work

By now we have discussed many revolutionary and thorough methods of measuring productivity and work in software engineering on a very large scale. However, it is important for us to consider the repercussions this kind of measurement could have also. When else in human history have, we compiled this amount of information about human labor? You could argue that given this is simply measuring the work an employee does that this is admissible but when company decisions start being made based on this information, that is when problems will begin to arise. We have already discussed how many of these measurements of information can be misleading without wider context that is sometimes not even measurable but to give an example:

Let us say there are budgetary concerns for a company, and it is decided that a software engineer must be let go. In the interest of the company and in the interest of fairness and avoiding bias, the manager decides he will decide who to fire algorithmically. There are numerous factors that could skew the data depending on the algorithm used but to give some examples: An engineer could be new to a particular software or language despite being much more capable in other areas relevant to the company and as such needs time to get used to this software or language; An engineer could be writing a lot more code than another but that code is mostly recycled from other areas in the project while the other engineer is doing much more challenging work; I could go on but the point stands that an algorithm will never be able to accurately access every possible factor behind an employees work history and as such could lead to some unfair consequences for engineers across the industry.

Conclusion

The realm of measuring software engineering is one that helps us as software engineers become more efficient, productive, cost effective and more as I hope I have demonstrated in this report. However, it is essential than when doing so that you not only consider the context of the measurements you are looking at but also the ethical implications of such practices.

Reading Material

All bullets are hyperlinks:

- **An Article By Robert Sidler On Software Productivity -** http://www.umsl.edu/~sauterv/analysis/488_f02_papers/SoftwareProductivity.html
- **An Explanation of The Various Work Items Found in Agile and How to Use Them in Azure DevOps -** https:/docs.microsoft.com/en-us/azure/devops/boards/work-items/about-work-items?view=azure-devops&tabs=agile-process
- **An Infopulse Article on 10 Software Development Metrics to Measure Productivity -** https://medium.com/@infopulseglobal_9037/top-10-software-development-metrics-to-measure-productivity-bcc9051c4615
- **An Article on The Agile Manifesto -** https://www.atlassian.com/agile/manifesto
- **Azure Dev Ops -** https:/azure.microsoft.com/en-us/services/devops/
- **Software Yoga Article on Cyclomatic Complexity -** https://www.softwareyoga.com/cyclomatic-complexity/
- **Software Engineering Productivity Measurement using Function Points: A Case Study -** https://journals.sagepub.com/doi/abs/10.1177/026839620001500108
- **Software Engineering Productivity: Concepts, Issues and Challenges -** https://www.igi-global.com/article/software-engineering-productivity/50541