Haste makes Waste. Observed and Used for Controlling Software Quality and Productivity

Keld Raaschou
SimCorp A/S, Denmark
keld.raaschou@simcorp.com

Abstract

The Rationale for Metrics

Some of the main reasons for performing metrics are that we want to support and improve quality and productivity. So which metrics can be more interesting than quantifications of quality and productivity themselves?

It is claimed that haste makes waste. But how hastily should we choose to work in order to obtain an optimum trade-off between quality and productivity?

Clearly, you are more productive when the quality of your work is high. But how much should the value of quality count?

To address these questions, the Danish software company, SimCorp, has established novelty concepts for quantification of haste, waste, quality and productivity.

Haste makes Waste

The measurements confirm the old saying that haste makes waste. Moreover, this observation has been used as a basis for a formula for productivity as a function of the changes made per hour and the costs per error.

Perspective

The perspective is that once you have determined the total costs per error, then you can consciously optimize and choose the right trade off between quality and productivity – simply by planning for the right level of changes per hour.

1. Introduction

SimCorp is a Danish software-company that delivers software for investment management. The product scope is our main product, "SimCorp Dimension", which is maintained in one large product line, mainly coded in APL.

A strict time-box based lifecycle model is used: every six months, a new, improved version is released. At any one time, a number of successive versions of the system are in use.

We have collected comprehensive metrics for the series of versions for more than 5 years. There is great stability, regularity and similarity between the versions. This means that the collected data provide an outstanding basis for meaningful and accurate metrics and comparisons between the properties of the versions. In addition, estimation and planning of the number of changes per version are well established.

One particular version is used for reference and has index=1 for any type of measurement. So if e.g. the quality of a particular version is 20% higher than the quality of the reference version, then its quality index is 1.20.

The symbol, $\propto$, indicates proportionality of relative measures, rather than true equality of absolute measures. This is no problem as long as you compare indexed values and look at trends. The square of a number is symbolized by "**2".

The scope for the formulas is the development organisation as a whole, and an entire version. E.g. Changes per hour is a measure of the changes made for that version by the entire development group, divided by the hours spent by the entire development group during the development period of the version.

2. Quality Concept

Quantification of quality does not make sense, unless you determine which view of quality that you want to quantify. SimCorp is aware of several different views of quality, but in this investigation, we address the error-related view of quality: a comparatively low number of customer reported errors in the versions that are released to customers.

"Comparatively" means that we do not just look at the absolute number of reported errors. We also take into account the amount of changes and the level of customer usage over its lifetime.

A further explanation is given in the following subsections.

2.1. Amount of Changes

We have managed to extract data for the number and type of code changes from one version to the next on a per code line basis. These measures have been weighted by our well-established estimation parameters for the effort required for the different sizes and types of change, i.e. additions, modifications or deletions.

The result is a measure of the amount of changes that were made in each version. The measure resembles the "code churn" measures reported by Nagappan et. al. [2]. But our measure is a weighted count instead of a simple count.

2.2. Inherent Errors

"Inherent errors" is the level of errors that customers would report during the entire lifetimes of the versions, if they were all exposed to the same, standardized level of customer usage. This calculation uses different weights for different criticalities of errors.

Via a special model for customer usage, we were able to predict and calculate the different versions' number of inherent errors.

In the formulas below, the weighted level of inherent errors is just referred to as "Errors".

2.3. Formula for quality

According to the quality view that we have chosen in this context, quality is the inverse of the number of errors per change, so:

$$\text{Quality} \propto \frac{\text{AmountOfChanges}}{\text{Errors}}$$

3. Productivity Concept

3.1. Rationale for a New Concept

According to IEEE-1045 [1], productivity is a measure of output/input, or: the produced result divided by the resources used for producing that result.

In our case, where our production is the changes that we make, you would traditionally say that:

$$\text{Productivity} \propto \frac{\text{AmountOfChanges}}{\text{DevelopmentEffort}}$$

IEEE-1045 [1] also recognizes that "productivity metrics should be interpreted in the context of the overall quality of the product". But it is outside the scope of the standard to determine how that aspect could be quantified.

Our suggestion is to try and quantify a fair measure of how productive the organisation really is: you should give credit to the quality of what you produce, and also to the difficulties associated with the effect of size and complexity of the system that you are maintaining.

3.2. Effect of Complexity

The isolated effect of an increased size and complexity of the system is that any typical maintenance task would require more effort. E.g. if it required 5% more effort to perform the changes in a new version, then we would say that the effect of complexity in the new version was 1.05, relative to the old version. By particular methods, we have been able to establish rough measures of the effect of the increasing size and complexity of the system.

Note that the effect of complexity should not be confused with traditional measures of the complexity itself.

3.3. Value of Quality

How should quality be taken into account? We cannot use the quality measure directly: twice the quality implies that you are more productive, but maybe not twice as productive. It appears to be **impossible** to obtain consensus about how the value of quality should be used in a productivity measure. But the following directions show how to circumvent this problem.

If it is impossible to compare the productivity of versions with different quality, then we are forced to compare versions at the point in time, where they have the same quality. Theoretically, this would happen at some future point in time, where the version has been subject to a standardized level of customer usage during all of its useful life. At that time, all the version's inherent errors would, by definition, have been found and corrected. So at this point, all versions would somehow have obtained the same quality. Therefore, at this point, they would also have the same value of quality. And this is how they can be viewed in a predicted state, where everybody can agree that productivity measures are directly comparable.

So you view the versions as they would be at a later point in their lifecycle. This also implies extra costs.

The extra costs are the foreseen costs of all the inherent errors that would eventually be found.

So now: *Costs = DevelopmentCosts + ErrorCosts*

3.4. Final Formula for Productivity

When we consider both the effect of complexity and the value of quality, the formula changes to:

$$\text{Productivity} \propto \frac{\text{AmountOfChanges} \times \text{EffectOfComplexity}}{\text{DevelopmentCosts} + \text{ErrorCosts}}$$

4. Definition of Haste and Waste

In the actual context, haste is a measure of the level of rush that developers feel when they perform changes in order to maintain software.

The more changes you make per hour, the more haste you have. The more complex and large your system is, the more haste you will have, trying to maintain it. The more skilled and efficient you are, the less haste you shall need. So:

$$\text{Haste} \propto \frac{\text{ChangesPerHour} \times \text{EffectOfComplexity}}{\text{AverageSkillFactor}}$$

Personal skill factors are a well-established and constantly tuned basis for estimation in SimCorp: For each individual, it is subjectively determined how efficiently that person is working compared to a standard staff with 1 year of experience. For the organisation as a whole, you can calculate the average skill factor.

Waste is defined as the relative costs of bad quality.

$$\text{Waste} = \frac{\text{CostOfErrors}}{\text{DevelopmentCosts}}$$

5. Comparison of Haste and Waste

Both measures, haste and waste, have been normalised by indexing so that the indices for a particular reference version were set to 1. You would expect that high values of haste would give low values of quality. If you assumed that Haste × Quality were constant, then, according to the definitions, you would expect that:

$$\text{Waste} \propto (\text{Haste}**2) \times \frac{\textit{AverageSkillFactor}}{\textit{EffectOfComplexity}}$$

In our case, the AverageSkillFactors have had approximately the same trend as the EffectOfComplexity. So you would expect a high correlation between Haste**2 and Waste. And indeed, the correlation coefficient between the indices for Haste**2 and Waste for 6 successive versions was very high, namely around 0.98. So this is new illustration of the old saying: “Haste makes waste”.

6. Use of Result for Control of Quality and Productivity

In theory, you could alternate between 2 different types of development periods: One where you made an extremely high number of changes per hour, and another, where you had little capacity, because you were tied up with corrections of errors made in the previous development period.

But let us, for the sake of simplicity, assume a steady state, where the relative error costs, the development costs, the skill factors and the effect of complexity were constant from version to version. For such a steady state, the haste-makes-waste rule could be used for an easy gross calculation on how to optimize productivity:

Due to waste, the effective development capacity would be the basic capacity minus the lost capacity. Moreover, in a steady state, the lost capacity would be equal to the effective capacity times the waste.

So, the EffectiveCapacity = NormalCapacity minus (EffectiveCapacity × Waste). This implies that our capacity for making productive work is reduced by a factor = (1 + Waste). So:

$$\text{Productivity} \propto \frac{\text{ChangesPerHour}}{1 + \text{Waste}}$$

If we assume that *Waste* ∝ *Haste* **2, then:

$$\text{Waste} = \text{Cst} \times \text{ChangesPerHour}**2$$

Where *Cst* is a constant, and:

$$\text{Cst} \propto \frac{\text{EffectOfComplexity}**2 \times \text{CostsPerError}}{\text{AverageSkillFactor}**2}$$

Therefore, by insertion normalizing so that the indexed values for the reference version become= 1:

$$\text{Productivity} = \frac{(1 + \text{Cst}) \times \text{ChangesPerHour}}{1 + \text{Cst} \times \text{ChangesPerHour}**2}$$

A smooth graph of indexed values for Productivity versus indexed values for ChangesPerHour was made, and the constant, Cst, that gave the best fit, assuming external costs = 0, was calculated.

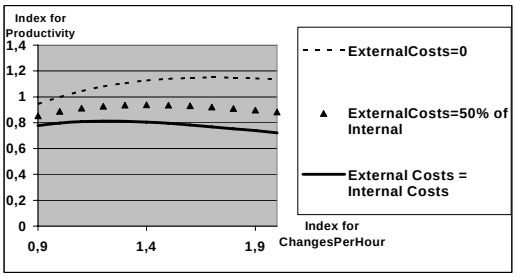

Figure 1. Productivity vs. ChangesPerHour

Figure 1 shows the productivity index relative to the state of the reference version if external costs were assumed to be = 0. Graphs for three different assumptions of the external costs are shown.

The graphs for Productivity as a function of ChangesPerHour resemble a number of upside down parabolas – one for each assumed value of costs per error. The costs per error = (Internal costs per error + External costs per error) determine which of the graphs that shall be used for optimization. The internal costs per error are well known. The external costs per error, like loss of goodwill, are harder to quantify. Quantification of equivalent external costs per error is a good candidate for future research.

For any assumption of CostsPerError, you can calculate the value of ChangesPerHour that will optimize the Productivity. Alternatively, you could choose and obtain some other desired trade-off between quality and productivity.

Note that the optimal number of ChangesPerHour is calculated using the assumption of a long-term steady state. Still, it is only valid for the current set of parameter values. If the effect of complexity or the costs per error increase, then the currently optimal number of ChangesPerHour will be reduced. On the other hand, if the organisation becomes more mature and efficient, then the currently optimal number of ChangesPerHour will increase, and the obtainable productivity will increase.

7. Lessons Learned

Via predictions of inherent error levels, we have established a new type of quality measure.

Also, we have managed to establish a novelty productivity concept, which is viewed as a fair measure of how productive we really are. The reason for this acceptance is that it also takes complexity and quality into account, and that there is a good explanation of why and how it is done.

The measures for quality and productivity have proven useful in several different fields: monitoring of trends, planning, awareness, motivation via visual progress, correction of wrong myths, measurement of what works well, and potentially as a basis for incentive systems.

Measures for haste and waste have been established, and a high correlation between Haste squared and Waste has been observed. This confirms our intuitive expectations. The observation has been used as a basis for a simplified gross calculation of a formula for optimization of productivity via choice of changes per hour. Our conscious choice of the number of changes in each version is now supported by these results.

These accomplishments are useful. But you will probably need a high level of reliable metrics and comparable projects, in order to fully utilize our results in other companies.

Everybody talks about the weather, the quality and the productivity. But who does anything about it? The described concepts support the perspective that quality and productivity are not just something that “happens”: it is something that can be consciously chosen by planning the level of changes per hour.

8. References

[1] *IEEE Standard for Software Productivity Metrics*, IEEE Std 1045-1992, 17 September 1992.

[2] Nachiappan Nagappan & Thomas Ball, *Use of Relative Code Churn Measures to Predict System Defect Density*, ICSE ‘05 , May 15-21 2005, http://research.microsoft.com/research/pubs/view.aspx?type=Publication&id=1359