SEPARATING AND QUANTIFYING VALUE AND WASTE TO IMPROVE OPERATIONAL PERFORMANCE IN SOFTWARE DEVELOPMENT

Darius Khodawandi
Wilhelm-Schickard-Institute, University of Tuebingen, Sand 13, D-72076, Tuebingen, Germany
darius.khodawandi@mac.com

Keywords: Software development productivity, Productivity quantification, Lean software development, Agile software development, Operational performance.

Abstract: Software companies choose to implement Agile software development practices to increase the frequency of their release cycles, stabilize quality, reduce cost - or in short: improve operational performance by increasing productivity. Yet quantifying, what productivity improvement the implementation of Agile software development resulted or will result in is still a big challenge. This paper introduces a new concept to quantify productivity based on the theory of Lean Management - which is closely related to Agile software development - and the specific concept to separate value and waste in processes and products. The author claims that consideration of three dimensions is necessary in the context of software development: (1) the value adding share of product features, (2) the share of the product which contributes to the overall feature set in a value adding way, and (3) the value adding share of time employees can spend when creating software. The ideas proposed in this paper are subject to ongoing trials in an industry environment, which the author is directly involved in. The overall contribution of this paper is a) a new concept to quantify productivity in software development and b) results from initial application of the concept in a large enterprise environment to quantify one of the three dimensions of the overall quantification concept.

1 INTRODUCTION

The market for On-Demand software solutions is growing, and companies are adjusting their business models accordingly. When looking at the changes required more closely, they are all opportunities which - from the authors' point of view - will lead to an industry-wide improvement of customer-orientation and productivity. By offering On-Demand software solutions, installing learning cycles that quickly turn insights from the field into redesigned solutions becomes more easily possible than when solutions are deployed in a traditional fashion. At the same time, productivity will increase because the process by which software is developed has to change dramatically. Based on the previous statement, the process must be capable of handling frequent learnings (i.e. new requirements or changes) and turn them into useable software in a reasonable timeframe. These capabilities are inherent to the concepts of Agile and Lean Software Development, since they promise shorter lead time from idea to customer use. One reason to be able to reduce lead time by implementing the concepts of Agile and Lean Software Development are productivity gains, which are achieved by e.g. assigning all relevant roles to one team, dedicating team members to only one team and one team to one task at a time, and by co-locating team members to make communication more efficient.

Even though it seems obvious that the concepts of Agile and Lean Software Development - of which only an extract was mentioned above - will lead to improved productivity, there are good reasons to try to quantify the effects of implementation. Besides the convincing nature of figures which is relevant from a pure organizational change view when implementing Agile or Lean Software Development, being able to quantify productivity improvement long-term is regarded as an essential capability for learning software companies. Especially when regarding the concept of Lean Software Development, it is an inherent objective to reduce waste within a company, i.e. any tasks or product attributes which the customer would not be willing to pay for.

Therefore the objective of this research is to determine a method to quantify productivity of software development from an enterprise perspective, which helps to quantitatively incentivize and prove that working in an Agile or Lean Software Development Model leads to overall improvements. This includes determining what should be measured, and how to measure it in a way to be meaningful and economically implementable. This research is not intended to determine a figure of what improvement to expect from implementing Agile or Lean Software Development, nor how to create a measurement method which is suitable for industry-wide benchmarking.

Section 2 will describe relevant background knowledge regarding Agile and Lean Software Development, and existing quantification approaches for productivity in the software industry. Section 3 introduces and describes the quantification approach in a sequence of *what* to measure, then *how* to measure, and finally draws conclusions. Section 4 contains next steps.

2 BACKGROUND

The relevant existing research for the scope of this paper can be divided into the fields of Agile and Lean Software Development, as well as Productivity Quantification. A more generic background is seen in the work of Drucker (1999) relating to knowledge workers.

2.1 Agile and Lean Software Development

The evolution of process models and concepts for software development reach back to the traditional waterfall and V-Models, followed by Extreme Programming, Spiral or RUP. More recently, approaches have emerged that claim to be more reactive to the environment and the nature of software development - rapid new insights leading to frequent changes, its non deterministic nature (maybe even as a typical attribute to development in general), and with respect to software development specifically, the seemingly indefinite solution space to a given problem space.

These recent approaches go by the name of Agile or Lean Software Development. Both approaches have large commonalities with respect to their basic motivation (e.g. accelerate delivery, build quality in, customer orientation) - and even though there are some differences (e.g. Agile does not explicitly ask for a separation of value and waste, nor does it define how continuous improvement should work), the author will refer to the two in combination for the purpose of this paper, as it makes no difference in this context. For further reading, Poppendieck (2008) is a frequently cited source for Agile and Lean Software Development

2.2 Productivity Quantification

Quantification of productivity is not a new challenge within the software industry (Scacchi, 1994). A commonly accepted approach is to put the input in relation to the output when calculating productivity. Input can be regarded as the resources utilized and output as what the result is worth. The most difficult aspect is the quantification of output, as it should not be a measure dependent on technical complexity or size (e.g. lines of code), but dependent on the "size" of the customer problem which it solves. The most commonly used and cited approaches are Function Points and User Stories and the corresponsing Story Points. These approaches are also utilized to estimate effort based on expected complexity of code.

The existing approaches do not yet fully fit the intentions of Agile and Lean Software Development in respect to their contribution to productivity improvement:

1. Instead of coding efficiency or speed, the question is how efficiently the knowledge worker e.g. a developer can use his/her available time in total (i.e. 100%) - this means what occupies the developer, e.g. rework or creative work
2. An aspect not considered at all in alternative productivity quantification approaches is the question, how much of the assets created by knowledge workers are actually important from the customer point of view.

In this paper, a concept is introduced which attempts to adress the above mentioned weaknesses of existing productivity quantification approaches in the industry. Parts of the concept have been applied in practice and are subject to further research.

3 QUANTIFICATION APPROACH

The first question this paper attempts to answer is *what* to measure. Compared to existing research, the author proposes to go down a different path than the popular approaches. Instead of a white-box view, the author proposes a more black-box oriented selection of criteria to measure. The reasoning for this is to avoid trying to replicate the complexity of a knowledge worker organization in its cause and effect relationships within a productivity figure.

3.1 Overall Concept

Considering the identified weaknesses of the existing approaches and also the strategies which Boehm (2007) identified to improve software productivity (get the best people, make development steps more efficient, eliminate development steps, eliminate rework, build simpler products, and reuse components), the following 3 dimensions should be considered to determine the value-adding share (from the customer point of view) and thus the productivity of software development: (1) feature value add, (2) product value add, and (3) process value add. The order chosen is based on the perceivable sequence of the dimensions from the customers perspective, has no further relevance.

The result of this quantification approach is an overall percentage, resulting from the multiplication of the individual dimensions' value adding share. The overall value adding share is an indicator for the operational performance of software development, as it considers the share of resources on the input side spent value adding, as well as a differentiated view on the output side by distinguishing the customer relevance and the product architecture.

A benefit of the chosen approach is seen in the fact that the unit is percent. Maybe one of the most trivial issues is that a ratio is hard to relate to from an employees perspective (what does a productivity improvement from 2.3 to 3.1 really mean?). The percent figure gives an easy to understand value. The next section describes the concept of separating value and waste focusing on dimension 3 (process value add), as this is the dimension for which trial results are already available.

3.2 Separating Value and Waste

The remaining paper focuses on *how* to measure productivity. The main question of dimension 1 - process value add is: how much of an employees time can be spent on tasks, which are of value to the customer? In order to measure this dimension, the disctinction of value adding compared to non value adding tasks needs to be clarified, as well as the level of measurement.

The most reliable distinction criteria for value add from non value add is - according to Lean Management theory: value adding activities are those activities, which the (end) customer is willing to pay for and thus should be maximized. Non value adding tasks are the remaining activities which should either be minimized (necessary non value adding - e.g. testing) or eliminated (obvious non value adding - e.g. rework). According to this definition, testing as a task is non value adding - a definition which is unusual at first sight but makes sense when one considers the business rationale not to maximize testing, but to try to achieve better quality from the customer point of view with less testing.

Regarding the dimensions (1) feature value add and (2) product value add, research is still required to determine adequate measurement concepts. Dimension (1) can be based on direct customer feedback, in a form where product features are listed and customers are able to rank or classify features (or feature groups) according to their perceived value. Dimension (2) can be based on the concept of target-costing or the quality function deployment method, by which the value of features from the customers point of view is determined and then their cost of creation is put into relation.

3.3 Results to Date

So far, three trials have been executed in an attempt to quantify the process value add axis. Figure 1 shows the task types and their categorization into the classes value add, necessary non value add, and obvious non value add.

	Categorization		
Task categories	Value Add	Necessary Non-Value Add	Obvious Non-Value Add
e.g. Code Development	X		
Creating reports		X	
Design compliance checking		X	
Fixing		X	
Handover and review		X	
Information Exchange		X	
Office Activities		X	
Project planning and setup		X	
Reviewing content		X	
Searching			X
Testing		X	
Travelling		X	
Waiting			X
Meetings		X	

Figure 1: Task categories applied in trials.

The first method applied to quantify the value adding share of the process is based on individuals submitting data on a daily basis, about which time was spent for which predefined task category in a granularity of 15 minute intervals. The application of this method was on-top to regular tasks of the day and for 5 days in a row to eliminate day-specific patterns (e.g. dedicated meeting or travel days). The submitted data was normalized in a way that the length of an individuals work day did not have an impact on the results, i.e. only the calculated percent-figures per individual where considered during aggregation. The second method applied is the integration of the above mentioned task categories into a time tracking system. This way, the generation of data was expected not to be on-top but part of the routine to track working hours. Also, the method allowed continuous generation of data. The third method applied was a simple daily indication, which requested from users their opinion on the value add share of the day. This daily submission was handled by a small tool, which collected the input in a database.

The trial application of the quantification concept follows the definition of measurement as "a quantitavely expressed reduction of uncertainty based on one or more observations" (Hubbard , 2010, p. 23), as it was rather pragmatic. The next section will elaborate on the learnings from the trials.

3.4 Evaluation and Conclusions

Overall, the following modifications were identified during the initial trials to be considered moving forward. (1) The duration of sampling should be aligned to the duration of development sprints. If sprints take e.g. 4 weeks, the duration to collect data should be at minimum 1 or a multiple of the sprint length. This is due to the fact, that otherwise the reported tasks would be biased due to the nature that e.g. at a beginning of a sprint there is more meeting time to discuss the expected outcome. The hypothesis moving forward is that data collection for one sprint is sufficient for a measurement, and that longer data collection would not lead to the equivalent of more informational value. (2) Measuring at the level of tasks may have disguised relevant productivity losses, which occur on the level of activities (i.e. inside tasks). Tadhani (1984) also suggests that limiting measurement to one level is insufficient. It is proposed to distinguish: the level of tasks and the (lower) level of activities. The necessity of this distinction is made clear by the following example: when considering the amount of time spent on testing, taking a closer look reveals that even if an employee spends 4 hours on testing, the actual net productive time is typically less due to necessary test infrastructure setup time, finding and loading of test cases, initializing the test - all activity level steps that need to occur before the actual test can be conducted. Moving forward, the hypothesis is that at least another 25% of productivity is lost on the level of activities, and that especially system performance is a major contributor for this. (3) Regarding the general measurement method, a comparison of the three trials conducted shows that an improved version of method one seem to be most promising. Method two (work hour tracking) seems to deliver better quality data (based on a bigger sample size and a continuous generation), but has turned out to increase daily complexity of data entry because the number and granularity of task categories is higher than before. Adding the level of activities will even further worsen this situation. Method three seemed to be a simple approach for a large sample size, but the informational value is insufficient (only the value adding share is determined, with no data on the share of time spent on non value adding activities).

4 NEXT STEPS

The measurement method to determine the process value add will be revaluated in another trial, incorporating the insights described above, first in a university environment, then in a large enterprise environment. In parallel, research to determine adequate measurement concepts for the feature and product axes will continue.

REFERENCES

Boehm, B. W. (2007), Improving Software Productivity. In Selby R. W., Software Engineering: Barry W. Boehm's Lifetime Contributions to Software Development, Management, and Research, John Wiley & Sons.

Tadhani, A. J. (1984). Factors Affecting Programmer Productivity During Application Development. In IBM Systems Journal, Vol. 23(1), 19-35.

Scacchi, W. (1994). Understanding Software Productivity. In Advances in Software Engineering and Knowledge Engineering, D. Hurley (Ed.), Vol. 4, 37-70.

Hubbard, D. (2010). How to Measure Anything: Finding the Value in Intangibles in Business (2nd ed.). John Wiley & Sons.

Poppendieck, M. and T. (2008). Lean software development: An agile toolkit. Addison-Wesley.

Drucker, P. F. (1999). Management Challenges of the 21st Century. New York: Harper Business.