Measuring Productivity in Agile Software Development Process: A Scoping Study

Syed Muhammad Ali Shah
SICS Swedish ICT AB
syed.shah@sics.se

Efi Papatheocharous
SICS Swedish ICT AB
efi.papatheocharous@sics.se

Jaana Nyfjord
SICS Swedish ICT AB
jaananyfjord@hotmail.com

ABSTRACT
An agile software development process is often claimed to increase productivity. However, productivity measurement in agile software development is little researched. Measures are not explicitly defined nor commonly agreed upon. In this paper, we highlight the agile productivity measures reported in literature by means of a research method called scoping study. We were able to identify 12 papers reporting the productivity measures in agile software development processes. We found that finding, understanding and putting into use agile productivity definitions is not an easy task. From the perspective of common roles in agile software development process and existing knowledge workers' productivity dimensions, we also emphasize that none of the productivity measures satisfy these fully. We recommend that future effort should be focused on defining agile productivity in measurable, practicable and meaningful form.

Categories and Subject Descriptors
D.2.9 [Management]: Productivity

General Terms
Measurement

Keywords
Agile software development process, productivity, measurement

1. INTRODUCTION
In recent decades, the software engineering discipline has seen the emergence of many new software development methods and processes. The emergence of new methods and processes requires relevant measuring methods for better visualization and control of software development. Consequently, new metrics are basically not needed in all areas, but only where the agile principles have an impact on how the project is managed and where the agile process is fundamentally different from a rich/classical process [1]. For instance, there are different metrics used for measuring the size of software entities. In the early 70's, the International Organization of Standards (ISO) standardized the first metric for measuring software size called 'function points' (FP). Different variants emerged over time such as COSMIC-FFP, IFPUG, MK II and NESMA, which also became common measures of size [2]. Another very common metric used by researchers and practitioners to determine the size of software is 'lines of code' (LOC). Both aforementioned size metrics have been criticized a lot by their users [3][4].

ICSSP'15, August 24–26, 2015, Tallinn, Estonia
© 2015 ACM. 978-1-4503-3346-7/15/08...$15.00
http://dx.doi.org/10.1145/2785592.2785618

In 2001, a philosophy for developing software called 'agile software development' was introduced. Agile software development is a group of software development methods, based on a collection of iterative and incremental concepts, principles and practices [5]. The agile software development manifesto [6] defines the essential principles of agile methods by valuing:

- Individuals and interactions over processes and tools,
- Working software over comprehensive documentation,
- Customer collaboration over contract negotiation,
- Responding to change over following a plan.

Agile comes up with a different philosophy of developing software requiring a set of different measurement metrics and methods, e.g., story points and planning poker, for measuring the complexity of requirements to satisfy the agile philosophy, and are different from traditional software development methods.

Considering the agile manifesto one such requirement could be to redefine productivity metrics in agile, as the agile software development process depends on different roles, ways of working, interactions and collaborations. Productivity in agile is a not yet a well-studied domain [7]. Consequently, we performed a scoping study to identify how productivity in an agile software development process is defined and measured. The study has provided us with a set of the productivity measurements and allowed us to have an early view about the need to define, update or redefine the existing productivity measures for an agile software development process.

The paper is organized as follows; Section 2 discusses the related work. Section 3 presents the research design of the study. Section 4 presents the results. Finally, results are discussed in Section 5 and the conclusion is presented in Section 6.

2. RELATED WORK
Productivity in traditional software development has been studied intensely and remains a controversial issue [7]. Productivity usually is the ratio of output (e.g., features, functions) to input (e.g., time, effort), in other words it represents "what is required to produce" [8].

There are some studies [9][10][11] that have used productivity metrics related to traditional software development process, i.e. lines of code, function points etc., to measure productivity in agile software development process. For instance, Petersen's work [12] on productivity mentions that agile ways of working is completely different from the traditional software development process and thus, using productivity measures from traditional software process is not very promising. Measuring productivity in agile is not harder or easier, but what is important is what to consider in the changed ways of working in agile when defining the agile measurement metrics [12].

Considering related work in agile software development, the concept of productivity is not very clear. Melo et al, performed an interview-based study with different agile teams regarding the concept of productivity. They found that in most of the cases, the traditional definition of productivity was quite different from the perceived one. More specifically, 'Timeliness', 'Quantity' and 'Quality' were perceived as factors for measuring productivity. In addition, 'Customer Satisfaction' was also identified as a criterion of measuring productivity [13]. Lui and Chan [14] performed a controlled experiment called repeat-programming which can facilitate the understanding of relationships between human experience and programming productivity, and found that productivity in 'pair programming' achieved higher quality within minimum time.

Summarizing the related work it is evident that productivity measures are either taken from the traditional software development process or are developed under subjective viewpoints depending on the context. Hence, it is not well-known, what are the appropriate productivity measures in agile software development. In addition, to the best of our knowledge, no systematic study exists investigating this topic. Thus, we identify a need of conducting an early study, defining the scope of this currently very little investigated area of agile software development.

3. RESEARCH DESIGN

For conducting this research, different research methods were considered, i.e., mapping studies [15], literature reviews [16] and scoping studies [17]. However the aim of this research was to review the literature which to date has received little attention to identify gaps in the evidence base. We chose to carry out a scoping study. The methods of mapping study and systematic literature review could be applicable, but concerns with the large study structure, e.g., in a mapping study trying to build a classification scheme and structure, and in literature reviews adopting a precise, transparent and explicit approach which includes a series of phases to ensure extraction of all relevant evidences, resulted in avoiding these methods. We followed the framework of Arksey and O'Malley for conducting our scoping study [17], which takes the process of dissemination one step further by drawing conclusions from existing literature regarding the overall state of research. We consider this instrument to be the most suitable in our case to investigate an area that has received little attention so far. There are five stages in the framework. We present the first four stages of the study in the current subsections, while the fifth stage, containing the results, is presented in Section 4.

3.1 Stage 1: Research Question Definition

Our aim is to identify productivity measures in agile software development in literature. Therefore we formulated one research question.

RQ1: How is productivity measured in the agile software development process?

Increased productivity is the most advocated benefit agile brings with it [18][19]. Hence, there may be multiple metrics to measure productivity in agile software development process. We aim at finding some evidence in terms of productivity measures.

3.2 Stage 2: Relevant Paper Identification

We considered papers published from 2000 to 2014. The search string used was ("Agile" AND "Productivity"). The search was applied through 'Science Direct', 'Springer Link', 'IEEE Xplore' and 'ACM' digital library. These are recommended software engineering research databases [16] and we believe they would mostly cover the search we were aiming for.

The website search function "search in abstract" was used. The number of papers collected per phase is shown in Table 1. As indicated, there are a number of "not available" papers. There could be many reasons for having such "not available" papers, such as, the paper is not freely available and requires purchasing of it, the paper is in another language than English or the paper only allows limited access (to look inside etc.).

Table 1 Distribution of papers

Source	Found	Downloaded	Not available	Included
IEEE Xplore	93	86	7	8
ACM	28	20	8	1
Springer Link	20	11	9	1
Science Direct	9	7	2	2
Total	150	124	26	12

3.3 Stage 3: Study Selection

We scanned all downloaded papers to find evidence in literature of measuring productivity in agile software development processes. This led to a selection of 12 papers in total, out of 124. The inclusion and exclusion criteria employed is defined below.

Inclusion Criteria: The inclusion criteria were applied at three subsequent levels. First, the titles were screened. They were selected if the title contained 'agile' and 'productivity'. Second, we analyzed the abstracts of the papers where it had to demonstrate some experience in agile software development concerning productivity compared to other factors, such as quality, cost and schedule. As a third step, we thoroughly read the papers and included only those studies which described/discusses at least one of the following:

- agile software development process
- productivity
- method to calculate productivity, or productivity metrics

Exclusion Criteria: The studies that did not satisfy any of the inclusion criteria were excluded.

3.4 Stage 4: Charting the Data

We used the simplest form of 'tables' for the data extraction and charting. First, we charted the selected papers that we obtained from the process as shown in Table 2. Next, we extracted the relevant productivity metrics from the selected papers as shown in Table 3.

Table 2 List of selected studies

ID	Publications
J1	Layman, L.; Williams, L.; Cunningham, L., Motivations and measurements in an agile case study, Journal of Systems Architecture, Volume 52, Issue 11, November 2006, Pages 654-667, ISSN 1383-7621
J2	Tarhan, A.; Yilmaz, S. G., Systematic analyses and comparison of development performance and product quality of Incremental Process and Agile Process, Information and Software Technology, Volume 56, Issue 5, May 2014, Pages 477-494, ISSN 0950-5849,
J3	Moser, R.; Abrahamsson, P., Pedrycz, W., Sillitti, A., and Succi, G., "A Case Study on the Impact of Refactoring on Quality and Productivity in an Agile Team," in *Balancing Agility and Formalism in Software Engineering*, vol. 5082, B. Meyer, J. Nawrocki, and B. Walter, Eds. Springer Berlin Heidelberg, 2008, pp. 252–266.
J4	Parrish, A.; Smith, R.; Hale, D.; Hale, J., "A field study of developer pairs: productivity impacts and implications," *Software, IEEE* , vol.21, no.5, pp.76,79, Sept.-Oct. 2004
J5	Athanasiou, D.; Nugroho, A.; Visser, J.; Zaidman, A., "Test Code Quality and Its Relation to Issue Handling Performance," *Software Engineering, IEEE Transactions on* , vol.40, no.11, pp.1100,1125, Nov. 1 2014
C1	Ramasubbu, N.; Balan, R. K., 2009. The impact of process choice in high maturity environments: An empirical analysis. In *Proceedings of the 31st International Conference on Software Engineering* (ICSE '09). IEEE Computer Society, Washington, DC, USA, 529-539.
C2	Abrahamsson, P.; Koskela, J., "Extreme programming: a survey of empirical data from a controlled case study," *Empirical Software Engineering, 2004. ISESE '04. Proceedings. 2004 International Symposium on* , vol., no., pp.73,82, 19-20 Aug. 2004
C3	Hu Guang-yong, "Study and practice of import Scrum agile software development," *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on* , vol., no., pp.217,220, 27-29 May 2011
C4	Williams, L.; Brown, G.; Meltzer, A.; Nagappan, N., "Scrum + Engineering Practices: Experiences of Three Microsoft Teams," *Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on* , vol., no., pp.463,471, 22-23 Sept. 2011
C5	Abrahamsson, P., "Extreme programming: first results from a controlled case study," *Euromicro Conference, 2003. Proceedings. 29th* , vol., no., pp.259,266, 1-6 Sept. 2003
C6	de Souza Carvalho, W.C.; Rosa, P.F.; dos Santos Soares, M.; Teixeira da Cunha Junior, M.A.; Buiatte, L.C., "A Comparative Analysis of the Agile and Traditional Software Development Processes Productivity," *Computer Science Society (SCCC), 2011 30th International Conference of the Chilean* , vol., no., pp.74,82, 9-11 Nov. 2011
C7	Sutherland, J.; Viktorov, A.; Blount, J.; Puntikov, N., "Distributed Scrum: Agile Project Management with Outsourced Development Teams," *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on* , vol., no., pp.274a,274a, Jan. 2007

Table 3 Productivity metrics

Study	Productivity Metrics	Knowledge Worker Force
J1	Lines of executable code / staff day	Team
J1	Function Points / staff month	Team
J2	Lines of code / person-hour	Team
J3	Lines of code / hours	Team
J4	Average number of unadjusted function points completed per unit of time	Development team of 2 developers
J5	Resolved issues / month	Per developer
C1	Lines of code / person-hours	Team
C2	Lines of code / hour	Team
C3	Lines of code	Team
C4	Lines of code	Team
C5	Lines of code / hour	Development team of 4 developers
C6	Functional size / effort	Team (scrum)
C7	Function points / months	Per developer

4. STAGE 5: RESULTS

RQ1: How is productivity measured in the agile software development process?

Table 2 presents the selected papers from the surveyed literature. The primary studies included 7 conference papers and 5 journal papers. Table 3 presents state of the art productivity measures in the agile software development process. Table 3 column 1 presents the index of relevant scientific studies from which the metrics are extracted, column 2 presents the extracted agile productivity measure, and column 3 presents the knowledge worker focus group that relates with the metric, i.e., knowledge work force role ranging from team to pairs to individual roles. We have identified in total 12 studies that have used productivity measures in the context of agile software development. The metric 'lines of code' is extensively used in the surveyed literature and outperform in terms of frequency any other productivity metric for the agile software development process. From the total of 12 studies, 7 studies have used 'lines of code' as a productivity measure, where 3 studies have used 'function points'. One study (J1) has used both 'lines of code' and 'function points' as productivity measures. The remaining one study (J5) has used 'resolved issues' as productivity measure. It is strongly evident that the aforementioned identified measures are the ones mostly used for measuring the team's productivity in agile development processes.

5. DISCUSSION

From the surveyed literature, we identify the following trends of measuring productivity in an agile software development process:

- The scientific literature included in this scoping study discusses productivity, but has not defined it in a measurable form (out of 124 studies we only identified 12 studies in which productivity is defined in a measureable form).
- Lines of code are mostly used in relation to all other metrics followed by function points for measuring productivity in agile software development processes. Surprisingly, 'lines of code' and 'function points' are used extensively for measuring agile team productivity.

It is evident that in order to better control and manage an agile software development process there is a need to define agile metrics in a measureable form. Concerning the use of 'lines of code' and 'function points' as prominent metrics for measuring productivity, we hypothesize that such metrics do not depict the true meaning of productivity. More specifically, it is not apparent that how much time or effort in planning, thinking and information etc., is acquired to develop one line of code or function point. Moreover, refactoring, an important practice in agile, usually results in reducing the lines of code [20], therefore more lines of code do not mean better productivity [21]. Consequently, considering the agile software development process where individuals work in a team for a common artifact, individual performance needs to be aggregated on the team level. In a process like Scrum for example, a team is defined as a set of people consisting of four main roles such as team lead (scrum master), members (analyst, developer, and tester), product owner and stakeholder (user, manager etc.) [22].

Mostly agile practitioners pride to be highly productive, responsive and more collaborative, therefore productivity to them is of real concern [23]. Current agile productivity measures are not suitable for all roles and they only concentrate on coding (development) [24]. One should better strive for balance on what should be measured and how much value the measure brings up. Moreover, roles in agile context are defined as highly 'knowledgeable' [13]. That is, a role in agile is one who applies theoretical and analytical knowledge, acquired via formal education and experience, to develop new products or services [25]. Because knowledge is complex and hard to evaluate, this may change the way we measure and understand the productivity of agile knowledgeable roles and what value it delivers [7]. In this concern, Ramirez and Nembhard [26] summarized very important dimensions defining the knowledge worker productivity and considering software development. Melo et al. [13] also extracted nine highly-related productivity dimensions as follows:

- Quantity. Accounts for outputs (quantities) and outcomes (quantification of qualitative variables such as customer and worker satisfaction).
- Cost. Accounts for profitability, costs, etc.
- Timeliness. Accounts for meeting datelines, overtime needed to complete the work, and other time-related issues.
- Autonomy. Accounts for independence and how many things a worker can do simultaneously.
- Efficiency. Accounts for doing things right. Refers to any task, even if it is not important to the job. The task is completing meeting all the standards of time, quality, etc.
- Quality. Accounts for how good the work is.
- Effectiveness. Accounts for doing the right things. Refers just to the tasks that are important to the job, even if they are completed without meeting standards of time, quality, etc.
- Project success. Accounts for overall result of work, considering decision-making, team interaction, communication, predictability, crisis management, documentation, transferability of work, etc.
- Customer satisfaction. Accounts for the fact that the product needs to add value to the customer's business.

The agile software development process is a knowledge creating process requiring a team effort with different competences represented by many roles to develop a software artifact. Knowledge is complex and difficult to evaluate, this may change the way we measure and understand the productivity within an agile context [7]. The authors believe that this paper is the very first in nature to study explicitly productivity in agile. However the next step is to connect the paper results with existing agile measurements, e.g. story points, t-shirt sizing, etc.

6. SUMMARY

In summary, we could state that the present productivity measures are not efficient enough to satisfy the requirements for defining productivity in agile software development. It is clear that defining agile productivity measures must consider the knowledge dimension. In the future, we have a twofold research direction, first we aim at defining measureable productivity metrics for different agile roles that would also satisfy the knowledge worker dimensions and cover all aspects (from requirements to delivery of working product to a customer) of agile development process.

7. REFERENCES

[1] A. Abran, A. Sellami, and W. Suryn, "Metrology, measurement and metrics in software engineering," *Softw. Metr. Symp. 2003 Proc. Ninth Int.*, pp. 2–11, Sep. 2003.

[2] J-M. Desharnais, A. Abran, and J. Cuadrado, "Convertibility of Function Points to COSMIC-FFP: Identification and Analysis of Functional Outliers," 2006.

[3] B. Kitchenham and E. Mendes, "Software productivity measurement using multiple size measures," *Softw. Eng. IEEE Trans. On*, vol. 30, no. 12, pp. 1023–1035, Dec. 2004.

[4] E.P. Morozoff, "Using a Line of Code Metric to Understand Software Rework," *Softw. IEEE*, vol. 27, no. 1, pp. 72–77, Feb. 2010.

[5] J. Nyfjord, "Towards Integrating Agile Development and Risk Management," Stockholm University, PhD Thesis 08-008, 2008.

[6] K. Beck, "Manifesto for Agile Software Development," *http://agilemanifesto.org/*, 2014. .

[7] A. Trendowicz and J. Münch, "Chapter 6 Factors Influencing Software Development Productivity—State-of-the-Art and Industrial Experiences," in *Advances in Computers*, vol. Volume 77, Marvin V. Zelkowitz, Ed. Elsevier, 2009, pp. 185–241.

[8] C. Ebert and R. Dumke, *Software Measurement: Establish - Extract - Evaluate - Execute*. Springer-Verlag New York, Inc., 2007.

[9] W.C. de Souza Carvalho, P. F. Rosa, M. dos Santos Soares, M. A. Teixeira da Cunha Junior, and L. C. Buiatte, "A Comparative Analysis of the Agile and Traditional Software Development Processes Productivity," *Comput. Sci. Soc. SCCC 2011 30th Int. Conf. Chil.*, pp. 74–82, 2011.

[10] A. Parrish, R. Smith, D. Hale, and J. Hale, "A field study of developer pairs: productivity impacts and implications," *Softw. IEEE*, vol. 21, no. 5, pp. 76–79, 2004.

[11] J. Sutherland, A. Viktorov, J. Blount, and N. Puntikov, "Distributed Scrum: Agile Project Management with Outsourced Development Teams," *Syst. Sci. 2007 HICSS 2007 40th Annu. Hawaii Int. Conf. On*, p. 274a–274a, 2007.

[12] K. Petersen, "Measuring and predicting software productivity: A systematic map and review," *Spec. Sect. Softw. Eng. Track 24th Annu. Symp. Appl. Computer. Softw. Eng. Track 24th Annu. Symp. Appl. Comput.*, vol. 53, no. 4, pp. 317–343, Apr. 2011.

[13] C. Melo, D. S. Cruzes, F. Kon, and R. Conradi, "Agile Team Perceptions of Productivity Factors," *Agile Conf. AGILE 2011*, pp. 57–66, Aug. 2011.

[14] K. M. Lui and K. C. C. Chan, “Pair programming productivity: Novice–novice vs. expert–expert,” *Int. J. Hum.-Comput. Stud.*, vol. 64, no. 9, pp. 915–925, Sep. 2006.

[15] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, “Systematic mapping studies in software engineering,” in *Proceedings of the 12th international conference on Evaluation and Assessment in Software Engineering*, Italy, 2008, pp. 68–77.

[16] B. Kitchenham, S. Charters, D. Budgen, P. Brereton, M. Turne, M. Turner, S. Linkman, M. Jørgensen, E. Mendes, and G. Visaggio, “Guidelines for performing Systematic Literature Reviews in Software Engineering,” Software Engineering Group School of Com puter Science and Mathem atics Keele University and Department of Computer Science University of Durham Durham, UK, EBSE-2007-01, Jan. 2007.

[17] H. Arksey, and L. O’Malley, “Scoping studies: toward s a methodological framework,” *Nternational J. Soc. Res. Methodol.*, vol. 8, no. 1, pp. 19–32.

[18] S. Ilieva, P. Ivanov, and E. Stefanova, “Analyses of an agile methodology implementation,” *Euromicro Conf. 2004 Proc. 30th*, pp. 326–333, 2004.

[19] L. Layman, L. Williams, and L. Cunningham, “Exploring extreme programming in context: an industrial case study,” *Agile Dev. Conf. 2004*, pp. 32–41, 2004.

[20] S. A. M. Rizvi and Z. Khanam, “A methodology for refactoring legacy code,” *Electron. Comput. Technol. ICECT 2011 3rd Int. Conf. On*, vol. 6, pp. 198–200, Apr. 2011.

[21] M. Solla, A. Patel, and C. Wills, “New metric for measuring programmer productivity,” *Comput. Inform. ISCI 2011 IEEE Symp. On*, pp. 177–182, Mar. 2011.

[22] S.W. Ambler + Associates. “Roles on Agile Teams: From Small to Large Teams,” 2014. .

[23] D.J.Anderson, “Stretching agile to fit CMMI level 3 - the story of creting MSF for CMMI® process improvement at Microsoft corporation,” *Agile Conf. 2005 Proc.*, pp. 193–201, Jul. 2005.

[24] R. unnalan, M. Shereshevsky, and H. H. Ammar, “Pseudo dynamic mtrics [software metrics],” *Comput. Syst. Appl. 2005 3rd ACSIEEE Int. Conf. On*, p. 117, 2005.

[25] P. F Drucker, *Adventures of a Bystander*. Transaction Publishers, 1994.

[26] Y. Ramirez and D. A. Nembhard, “Measuring knowledge worker productivity,” *J. Intellect. Cap.*, vol. 5, no. 4, pp. 602–628, 2004.