Comprehensive Measurement Analysis for Software Productivity

Samer I. Mohamed

Department of electrical communication engineering, faculty of engineering, October University of Modern Science and Arts MSA (Egypt)

E-mail: saibrahim@msa.eun.eg

ABSTRACT

Software development productivity is one of the major and vital aspects that impacts software industry and time to market of many software products. Although many studies have been conducted to improve the productivity measurements within software engineering research domain, productivity is still an issue in current software development industry because not all impacting factors and their relationships are known. This paper sheds a light on some of these factors and assesses their impacts as seen by random sample of industrial software SMEs. It also elaborates the main best practices that help in improve the software productivity based on real industrial projects. The resulting list of factors and best practices can be utilized to guide further productivity analysis and be taken as basis for building improved and more optimized productivity models. Paper also identifies some of the productivity measurements challenges and recommends set of best practices that can be utilized as basis for productivity measurements and estimation models.

Keywords: Software Productivity, Volatility, Technical Factors, Non-technical Factors, Best Practices, SMEs, SMART Requirements, Software Metric.

1- INTRODUCTION

Software development industry nowadays becomes one of the main industries that contribute on the evolution of the computer-based systems. Many organizations currently investing huge amount of money to improve their productivity and time to market to gain larger market share and increase their operational margin. Productivity in software development has been an important research area for several decades now where successful organizations focus their R&D to improve.

There are many different measures for software productivity within the literature. The most common and traditional approaches are the lines-of-code (LOC) and function points (FP), i.e., the amount of LOC or FP produced per hour by a developer [1]. Based on this, there is a large amount of studies on various aspects of productivity. The two mentioned measures and several more dimensions have been analyzed and detailed within the literature.

Our contribution through this paper, is the introduction of a balanced and mixed approach for both the industrial and theoretical perspectives of those factors that impact the software productivity. Although the software engineering literature in that area often has a strong emphasis on technical factors such as the software size or the product complexity. However, there are other non-technical factors that impact software productivity as has been proved by Brodbeck [2] who has shown that more than a third of the time a typical software developer is not concerned with technical work.

A productivity measure commonly is understood as a ratio of outputs produced to resources consumed. Experience shows that no single productivity measure applies in all situations for all purposes. Instead, organizations must craft productivity measures appropriate to their processes and information needs. In addition to the wide range of possible inputs and outputs to be measured, the interpretation of the resulting productivity measures may be affected by other factors such as requirements changes and quality at delivery.

There are different standards for productivity measurements like IEEE 1045 standard which describes the calculation of productivity in terms of effort combined with counts of lines of code or function points. Besides ISO/IEC 15939 standard which is the basis for the Measurement and Analysis Process Area of the Capability Maturity Model – Integration.

Challenges around the productivity measurements arise because productivity may vary across the organization itself due to changes and dynamics within project itself with respect to other running projects. Besides, factors that impacting different projects are themselves different innatures.

The paper is organized as follows: section II gives a background for the early studies done on software productivity as a concept; section III provides overview of the productivity metric design best practices and highlight productivity measurement challenges; section IV introduces the case study targeted by this paper; where case study description, details, results, and recommendations are detailed; section V is the conclusion of this study.

2- BACKGROUND

Software development is a great expense for most organizations, thus, software development productivity can have a significant impact on the organization’s ability to compete and survive. Currently, most software development organizations are not optimized. There is an increasing demand for software especially for embedded systems. However, without improved efficiency, it will be difficult to take advantage of these opportunities in a cost-effective manner.

Tools will not be the only facility to succeed; but a need for a process that ensures quality software can be produced consistently and efficiently has an important effect. Like the various automobile manufacturers, different development organizations today typically have access to roughly the same production tools and technologies. The organizations that have a process for leveraging them most successfully are the ones with the highest productivity and the lowest production costs and the best one to compete.

There are extensive researches in the measurement of the development productivity. Humphrey and Singpurwalla [3] use the statistical techniques of time series analysis to predict the productivity of software development with reasonable accuracy. Blackburn et al. [4] imparts a global survey of software developers on improving speed and productivity of software development. The most famous model that involves productivity is COCOMO by Boehm [5], [6]. It is a cost-estimation model in which the productivity of the developers obviously plays a decisive role. Lakhanpal [6] concentrated on characteristics of groups and their influence on productivity. Brodbeck describes in [8] that in a survey, the projects with a higher communication effort also were more successful.

Even the intensity of internal communication is positively correlated with project success. This is in contrast to common software engineering belief that high communication effort hampers productivity. Wohlin and Ahlgren have described factors and their impact on time to market in [9]. They use 10 different factors in their study, mostly factors that are covered by the different publications. They also include product complexity, methods, tools and requirements stability that could be considered as technical factors.

Blackburn, Scudder, and VanWassenhove [10] studied the factors and methods that improved productivity in Western European companies. They found project duration and team size to be significant. Chatzoglou and Macaulay [11] interviewed participants of over a hundred software projects about several factors and their influence on productivity. They found that experience, knowledge and persistence of the team members is considered important. Also the motivation of the users and their communication with the rest of the team play a role. Finally, the available resources, tools and techniques used and the management style are important factors

These studies focus mainly on the measurement of the productivity, there are unfortunately very few investigations on the elements that influence the productivity. While the basic model for productivity measurement based on process that converts inputs into outputs consuming resources to do so. The input may be the requirements or cost invested for the software project and the product output may be another work product like documentation or value gain from the software product.

3- PRODUCTIVITY METRIC

3-1 METRIC DESIGN CONSIDERATIONS

Designer of any productivity measure should consider the following items through defining a precise productivity metrics:

- Scope of resources – which resources get counted?
- Scope of inputs (Efforts) – Which input efforts get counted?
- Scope of outputs (product) – which products get counted?

In the previous discussion we discussed couple of the basic sizing measures for the productivity output or numerator which are Function Point (FP) or functional input size measure and Source Line Of Code (SLOC) or physical output size measure. While there are factors impacting the efforts required to produce a given quantity of software (size) like smart technologies, code generator tools, and other non-technical aspects. Consequently the effects of these technologies must be considered in determining productivity either by weighting the size measures or defining multiple productivity measures for different development scenarios. More than one size measure may be needed to capture all of the information needed about the quantity of product delivered. That is software produced by different methods may need to be counted separately. Key through designing a metric to measure or judge about software productivity is to understand what the metric will size or measure and if the data required for that purpose is available and can be easily collected within the software organization. Each metric matters to specific team or someone based on value gained from the metric itself like governance or compliance requirements. For example size (SLOC, FP) and speed (Velocity, story points) metrics are important to project managers through planning, while quality and reliability metrics are important to organization top management and customers to maintain margin and revenue. It’s important to utilize the productivity metric through comparisons either between teams or over different period of times for the same team to measure the improvements gained from some planned actions.

The software engineering industry is domain where stakeholders, clients and the end users influence inputs and outputs, which produces a contribution to both the internal and external efficiency. Hence, a totally different approach to productivity has to be undertaken in order to obtain a global measure that establishes how well a software engineering organization uses resources to create outputs with acceptable perceived quality and customer value [28]. Thus, inputs and outputs measurement should consider both quantity and quality. This importance is reflected in the premises that Grönroos and Ojasalo established: “The better the perceived quality that is produced using a given amount of inputs (service provider’s inputs and customers’ inputs), the better the external efficiency is, resulting in improved service productivity” and “The more efficiently the service organization uses its own resources as input into the processes and the better the organization can educate and guide customers to give process-supporting inputs to produce a given amount of output, the better the internal [12].

3-2 METRIC DESIGN CHALLENGES

The challenge behind productivity metric is the multiple factors that impact the productivity outputs and inputs. One of the other main challenges with productivity metric design is the abstract level where the designed metric is applicable under different conditions and in different originations. This proved to be very challengeable especially when each organization has its own structure, environment and process aspects that are in total impact their productivity measurements. Organization maturity in measuring and collecting the metric data is one of the other factors that controls how the measurement process will be successful. Since software product development life cycle go through different phases starting from requirements elicitations towards delivery, you have to use different metrics to measure the productivity in each phase which adds more difficulty in tracking and data collections. One of the main and important challenges that highlighted within this paper is the impact of non-technical factors on the productivity measurements. Software engineering activities are capital intense, so the human factor has to be analyzed in any management practice order to obtain a more adequate result. In the context of productivity measurement, it is well accepted that factors related to personnel such as (technical, non-technical) capabilities and skills, and (programming language, project, process…) experience influence directly on productivity results. In addition to these factors, and considering the lack of literature related to this area, its recommended that other factors such as motivation, performance management practices, compensation and rewards systems, organizational climate, and happiness could influence productivity results; but it is not clear how they influence and how to introduce them in productivity measurement. Thus, a wide range of research possibilities presents through the combination of knowledge of human resources management and productivity management, which could lead to a transfer of cognition for a common research purpose [13]. Challenges related to designing metric for code reusability still under research on how it can be linked to productivity measurement. Besides the challenges related to unresolved links between code reuse and some other tasks in the software engineering cycle like requirements engineering and design phases make it hard to select Commercial of the Shelf (COTS). Another challenge is the design of metric that could be applied in both new development and maintenance projects, considering the differences.

3-3 PRODUCTIVITY METRIC SAMPLES

Once the inputs, outputs, and factors influencing productivity measurement are defined, a formulation of the measure can be established. Hence, specific metric can be defined and each organization may use one or several of them for measuring its productivity. In order to sum up the state of the art about inputs, outputs and metrics, the most used in each category are presented in Table 1. They are ordered according to the measurement difficulty, from easier to harder. The degree of representation scale of the production process itself is also represented: lower difficulty measures are less representative of the production process than harder measures.

Table 1. Samples for productivity inputs, outputs and metrics

Inputs	**Outputs**	**Metric (P=Productivity)**
Wages	Sales	P = Sales / Wages
Effort = Men Hours	TLOC = SLOC + DLOC	P = TLOC / Effort
Effort = Men Hours	Function or Feature Points (includes all the variations of the original ideal)	P = FP / Effort
Effort = Man hours	story point	P = #storypoint/Efforts (Agile development)
Multiple inputs	Multiple inputs	Data Envelopment Analysis (DEA)(i.e. Mahmood et al., 1996), capable of being used with any input-output measurement
Multiple inputs	Multiple inputs	Multifactor metrics (i.e. Kitchenham & Mendes, 2004), capable of being used with any input-output measurement
Multiple inputs	Multiple inputs	General Linear Model metrics, capable of being used with any input-output measurement

4- CASE STUDY

4-1 CASE DESCRIPTION

The Case is based on an industrial survey performed among group of 50 software engineers and Subject Matter experts (SMEs) from different industrial domains within the software development field. The selected sample of SMEs takes into consideration different diversity aspects within technology, industrial domain, SME job level, application domain, project types and software development models. This is basically to ensure unbiased outcomes and normal weight distribution of the different factors that impact software productivity within software development spectrum.

4-2 CASE DESCRIPTION

The survey presented in this case study has different types of questions varies between multiple choice questions MCQ and open type questions where interviewee has to put his own answer. Although 95% of the questions are MCQ but the remaining 5% were needed to assess interviewee judgments on some productivity factors and best practices.

The survey main objectives are basically:

- Gather basic information about interviewee and projects types.
- Assess time wasted in non-productive tasks compared to other productive ones.
- Evaluate those factors that impact the total productivity from interviewee perspective.
- Identify those best practices to improve the software productivity either adopted or proposed by the interviewee.
- Measure the impact of external and non-technical factors on productivity.

The design of the survey was done to assess ratio and impact of different factors impacting the productivity either technical or non-technical aspects.

Technical aspects like requirements volatility, tooling, technical training, rework due to poor quality and bug fixes, innovation support, project duration, application complexity, technical experience, status updates/admin impact, and modern programming practices have been assessed.

Non-technical aspects like appreciation and motivation, team cohesion, software size relative to application size (diseconomy of scale), turnover/attrition, work location, environmental effect like noise/lighting effect, defensive management, team size and roles and responsibilities clarity has been assessed and compared against technical aspects.

Each factor impact on productivity from the above listed ones has been assessed in range from low level to very high level.

Survey also has identified the best practices to improve the software productivity and the adoption methodology ranging from unknown level to standard level as follows:

- Desks away from loud employees like managers, support, sales that are always on the phone.
- Deal with SMART requirements.
- Improve estimation accuracy.

- Use short task schedule.
- Being part of small and well organized project team.
- Use prioritized task list.
- Less context switching between multiple projects, or because of changing specs.
- Make sure to allocate time for Refactoring and optimization before QA gets to it.
- Code reviews.
- Reusability.
- Technical training.
- Pairing between developers through development.
- Adopt minimal constrains validation.
- Improve communication between Business Side and developers.
- Eliminate scope creep.
- Co-operative work environment.
- Have a system for distributing tasks.
- Increasing code knowledge.
- Prevents customer-architect misunderstandings by supporting agile development processes with prototyping, short iterations, and other practices that promote early and frequent customer interaction.
- Prevents architect-developer misunderstandings by enforcing policies such as requiring that a test case be written for every use case, forcing developers to think about each requirement from different perspectives.

4-3 CASE RESULTS

In this section we will show the outcomes from the productivity survey and how these outcomes related to previous analytical studies in this field [14].

These outcomes will be classified into two main categories:

I) Impact of different factors on software development productivity either technical or non-technical aspects.

II) Best practices adopted by developers either related to technical or process/project-related aspects.

Table 2 shows the impact of technical factors on software productivity. Each factor range between 'Low' and 'High' through 'Average' values.

Table 2. Impact of technical aspects on software productivity

Factor/criteria	**Low (%)**	**Average (%)**	**High (%)**
Volatility	0	27	73
Tooling/training	13	33	54
Rework	40	33	27
Status updates	27	40	33
Innovation	27	20	53
Project duration	13	60	27
App. complexity	13	47	40
Technical experience	7	20	73
Modern programming practices	0	13	87

Data of Table 2 is illustrated graphically in Figure 1.

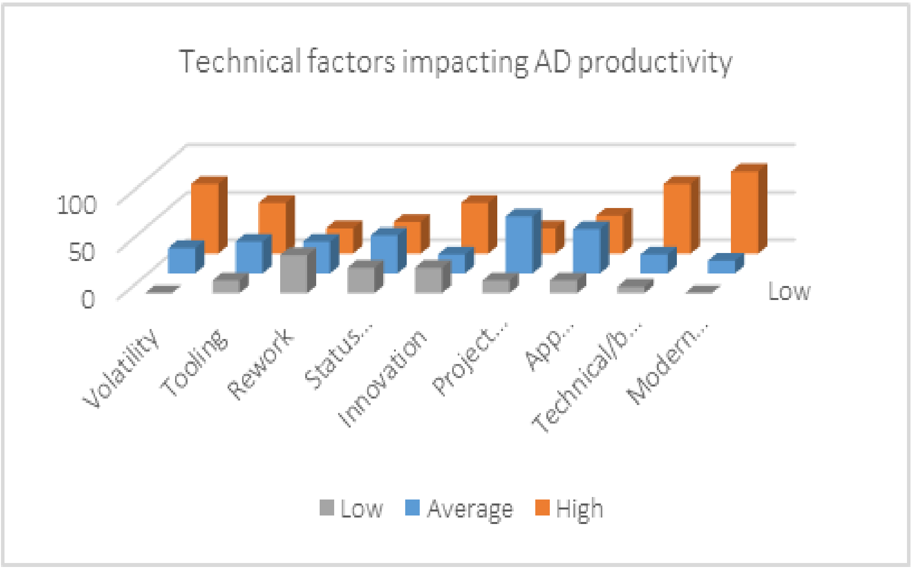

Figure 1. Technical factors effect on software development productivity

Table 3 shows the impact of non-technical factors on software productivity. Each factor range between 'Low' and 'High' through 'Average' values.

Table 3. Impact of non-technical aspects on software productivity

Factor/criteria	**Low (%)**	**Average (%)**	**High (%)**
Appreciation	0	20	80
Team cohesion	0	13	87
Software size	0	60	40
Turnover/attrition	0	20	80
Work location	7	7	86
Environmental effect	20	27	53
Defensive management	0	13	87
Team size	20	67	13
Roles and Responsibilities clarity	0	0	100

The data from Table 3 is also presented graphically in Figure 2.

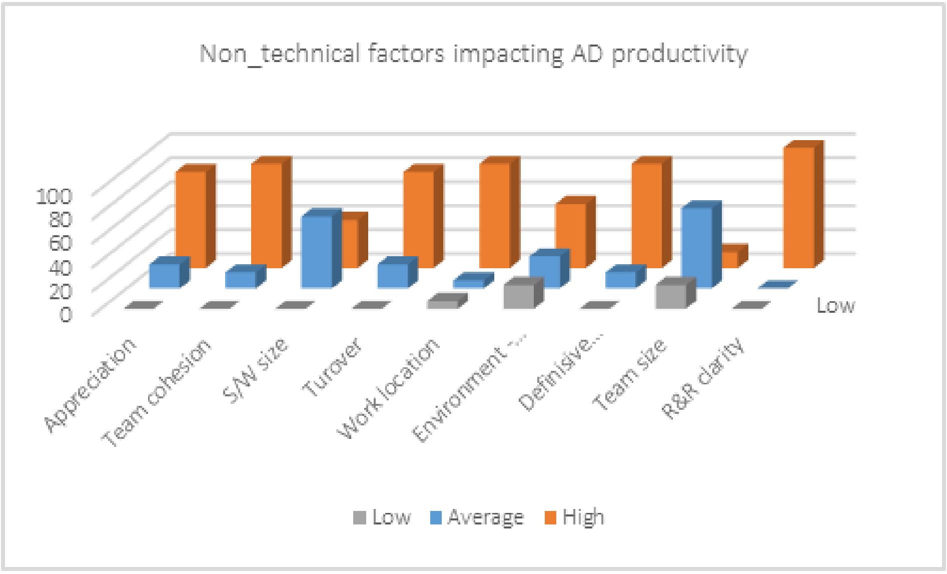

Figure 2. Non-Technical factors effect on software development productivity

Best practices adopted by software development industry have impact on the productivity as provided by the results/outcomes from the survey. These results are classified into main classes. The first class related to technical best practices while the second one is related to the project/process best practices.

Table 4 shows the impact of adopting different types of technical best practices on software productivity. Each practice range as detailed earlier between 'Not available' and, 'Standard' where practice is used as standard use, 'Training needed' where practice is used but need more development/improvement to be materialized, 'Has major value' where the practice has practical value on software productivity.

Table 4. Technical best practices impact on software productivity

Best practice	**Not available (%)**	**Stand-ard (%)**	**Training needed (%)**	**Major value (%)**
SMART requirements	60	27	13	0
Use Refactoring	67	33	0	0
Code reusability	13	33	27	27
Code reviews	13	80	7	0
Increase code knowledge	13	87	0	0
Agile development	33	53	7	7
Test case per use case	53	47	0	0
Technical training	0	7	27	66
Pairing	0	53	20	27
Have Business experi-ence	0	20	40	40
Minimal constrains vali-dation	0	13	67	20

The data from Table 4 is illustrated graphically in Figure 3.

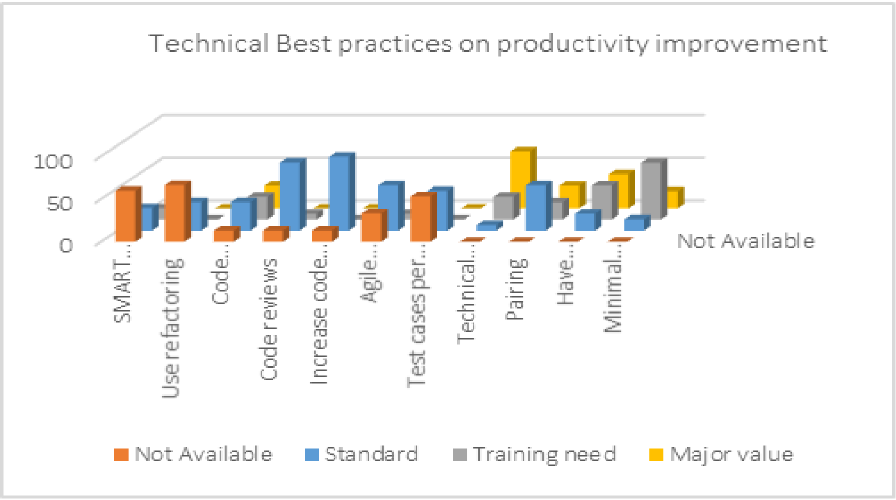

Figure 3. Technical best practices effect on software development productivity

Table 5 shows the impact of process/project best practices on software productivity. Each practice range as detailed earlier between 'Not available' and 'Standard'.

Table 5. Impact of process/project practices on software productivity

Best practice	Not availa-ble (%)	Standard (%)	Training needed (%)	Major value (%)
Desks away from noise	73	27	0	0
Less project switching	13	87	0	0
Communication improvement	40	33	27	0
Scope creep elim-ination	0	100	0	0
Improve estimation accuracy	13	73	13	0
Use task distribu-tion system	33	67	0	0
Use prioritized task list	13	87	0	0
Co-operative envi-ronment	0	53	20	27
Small project team	0	60	20	20
Short task sched-ule	0	33	33	34

The data from Table 5 is also presented graphically in Figure 4.

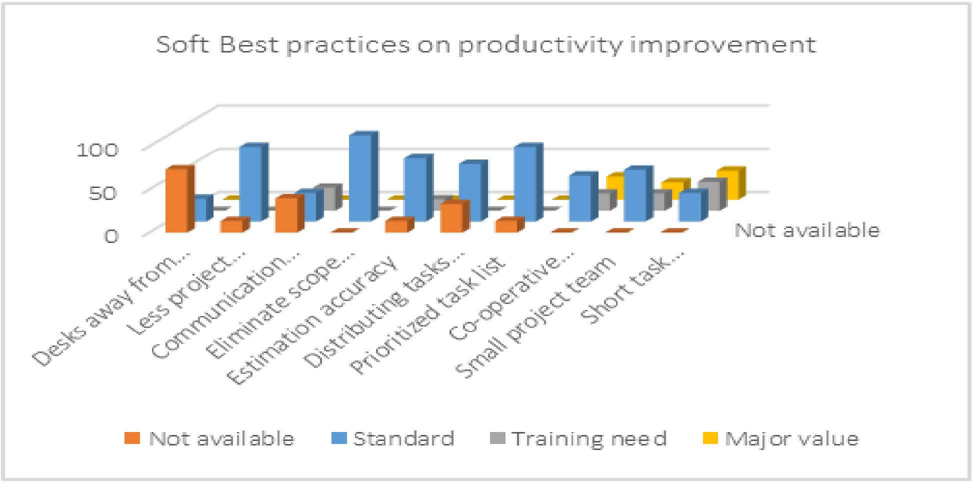

Figure 4. Process/project practices effect on software development productivity

4-4 CASE RESULTS ANALYSIS

By analyzing the results of this case study/survey, we conclude the following points as follows [15]:

With respect to technical factors, here are the analytical outcomes:

- Requirements volatility has major negative impact on productivity especially through the system design and development.
- Using state of the art tooling on different process, project, technical or technological levels will highly affect positively the software productivity through automation.
- Technical training is vital for productivity improvement and has high impact to improve the productivity.
- Rework due to poor quality or bug fixes impact productivity negatively but with lower weight/impact.
- Spending much more time on doing status updates/or admin work impacts productively negatively with average weight.
- Innovation support from the upper management has high positive impact on improving the software productivity.
- Project duration has average impact on the software productivity if it's within allowable limits (1 to 2 years). Productivity for those project with duration more than 2 years decay with time as developer's interest and motivation reach a saturation levels.
- Application complexity is directly correlated with the software productivity, but most of the interviewees see this has average impact only because the impact will be high at project start then decay with learning curve improvement along with project lifetime.
- Having business and technical experience with the application domain/field will help indeed to improve the development productivity and increasing code knowledge.
- Adopting modern programming methodologies have very high positive impact on improving software productivity [16].

With respect to non-technical factors, here are the analytical outcomes [17]:

- Appreciation and saying THANK YOU has the magical impact on the developer's spirit and motivation level, the thing that improves the software productivity [18].
- Team cohesion and healthy work environment have very high positive impact on software development productivity.
- Software size has neutral impact on the software productivity especially with adopting the latest state of the art methodologies and modern programming practices and tooling.

- Turnover/attrition has high negative impact on the software productivity because it's directly correlated to the productivity of the team in general and developer's spirit in particular.
- Having work location nearby home has a major impact on the developer productivity, the nearer the work location the more productivity outcomes and vice versa because of time/efforts waste due to lengthy transportation.
- Work environment conditions (Noise, lighting, seating, fresh air, others) have average impact on the productivity.
- Relationship between management and employees has high impact on the employee productivity, defensive management is negatively impacting resultant productivity and vice versa for supportive management.
- Working in a team or small groups indeed has major impact on productivity as explained in the team cohesion factor, but the team size also has an average impact on the productivity depending on the team size itself. If the team size up to five, then healthy communication and controllable/manageable deliverables can be maintained. Increasing the team above five will drastically impact communication in between developers and accordingly the resultant software productivity [19].
- Having clear roles and responsibilities for all stakeholders within the project will maintain healthy communication and clarify the boundaries between interacting roles, the thing that minimize the root cause for any conflicts or major escalations. This will definitely will improve the net productivity

With respect to technical best practices, here are the analytical outcomes [20]:

- Dealing with SMART requirements through the full development life cycle starting from elicitations towards testing through design and development has vital role to ensure a match between what is requested by the client and what is implemented by the development team. Although most of the interviewees see that adopting this practice is not available in many projects due to variations in maturity levels of the different stakeholders, but still acknowledge its vital value [21].
- Code refactoring/reusability is one of the other important aspects in modern programming practices to best reuse/re-structure the exiting code core without changing the external interface with the integrated systems. Survey shows that this practice has major value on improving the system performance/maintainability but not available for new development projects where time constrains exists [15].
- Code reviews is one of the important aspects in software development and adopted as standard by most of the interviewees through pairing approach. This ensures better quality and early bug detection which

improves the rework cost and enhance net productivity and accordingly the time to market [22].

- Agile development is one of the modern techniques for software development that adapts with the current market demand dynamics and fulfill the increase demand on new products with minimum time to market especially for mobile and small scale products. This practice is used as standard between most of the interviewees currently to adapt with market trends [15].
- Testing is one of the main components in the development life cycle. Incorporating the testing early in the design phase or even in the elicitation phase is very important to ensure every developed component/use case has its own test case. Adopting this as standard will lead to better quality, less rework costs, high productivity.
- Technical training and ongoing courses that adapt with the latest state of the art technologies is of a great need between most of the interviewees because it keeps them with the technological rapid advances and keeps the momentum for improving the business and technical experience.

With respect to process/project best practices, here are the analytical outcomes [21]

- Controlling the noise level in most of the organizations is very hard although it is important to facilitate a noise controlled climate for the software engineers. Most organizations currently balance between increasing number of meeting rooms versus the open space work locations depending on the development approach (ex. Agile development requires special arrangement for seating). Most of the interviewees see that 10-20% of their time almost wasted due to high noise level within the work place.
- Project management has an important role to facilitate structured and well organized climate for the development team to deliver starting from requirements specification towards project delivery, and to isolate any road blocks or management issues that waste their time. Switching between projects is mixed blessing as seen by most of the market leaders and management where it motivates the engineers while impacting the net productivity because of additional overhead to gain the knowhow and learning curve.
- Communication consumes basically considerable amount of anyone time, while it takes around 90% from the project managers, it also consumes between 10-30% of developer bandwidth. Thus improving the way of communication both internally between team members and externally with the project stakeholders has vital role and directly correlated with the team productivity in general and software engineer in specific [23].

- Scope creep is one of the aspects that lead to project failures and client dissatisfaction. Thus eliminating the scope creep is one of the standards adopted by many organizations to ensure project successful delivery.
- Improving the process and using more automation is vital to minimize the unnecessary overhead and admin work especially this is related to estimation process and other progress/reporting tracking tools. Although it's important for the PM to track the project progress and prioritize the task list for the team, S/he needs to control the additional overhead that impacts productivity via tooling and improved process.
- Working in a team is much better compared with working individually if it's performed within controllable ranges. Groups of 3 to 5 engineers is the optimum within software development teams from communication, cooperation, controllability, and delivery perspectives [24].

With respect to %time wasted in non-productive tasks relative to other productive ones, here are the analytical outcomes

- Meeting/talks (Technical) consumes around 20-30%.
- Presentations (Business related) consumes 5%.
- Project management/organization consumes 5%.
- Application development consumes 50%
- Others (ex. Coffee, Lunch) consumes 10%
- Most of S/W engineers consumes around 75% of their annual leaves.

Only 25% of the S/W engineers spend overtime hours outside of their normal working hours.

4-5 CASE RECOMMENDATIONS/PROPOSED ACTIONS

From the analytical outcomes detailed in earlier sections, we come up with list of recommendations and actions to better improve the software development productivity for any software development organization in general and software engineer in particular [25].

- Minimize the requirements volatility via adopting proper change/release management process, proper requirements management tools and modern/agile development methodology.
- Increase automation and tooling that eliminate manual and unnecessary overhead. Open source tools spread over the web have vital value and provide quick solutions for many problems with min/no costs.
- Organizations need to invest in their teams via technical training, R&D support, and innovation funds.

- Healthy relationship between the management and employees/engineers is the quick win for improving the net outcomes from the factory. This can be expressed in many ways simply via THANK YOU.
- Facilitate nearby location to home with multiple sites and WFH (Work From Home) facility will help in first minimize transportation time/efforts and improve the productivity.
- Organizations need to balance between the work from office and WFH days to maintain healthy communication and minimize the wasted times/efforts.
- Clear R&R is vital for the whole deliverables of any organization in general and software engineers in specific, where boundaries and interface with external world identified. This will enable the engineer to understand clearly what to do and what to avoid lead to better outcomes and minimal issues/escalations.
- Using modern programing practices like code refactoring/reusability, reviews, SMART requirements, agile development, pairing, and minimal constrains validation are important form the technical perspective to improve the quality, controllability and productivity.
- Adopting better process and project management policies like eliminate scope creep, facilitate cooperative environment, short task schedule, small project teams, prioritized list of tasks, distribution task tools, and desk away from noise sources will help in minimize the overhead and eliminate any sources of distractions.
- Designing productivity metric should be in sync with the organization structure and objectives. Where each metric/s should be associated with specific goal to be measured against regularly to track the progress and achievement of each team/project.
- Having concrete and well-established historical data system is vital for building productivity models based on correlations and analytics with the previous gathered data. This will help not only in improving the productivity measurements but also fine turning the model itself to better predict the growth and throughput improvement trends. This is directly linked with the organization profit and financial growth targets, where each organization in current harsh market conditions do whatever it takes to survive and gain more market share.
- Since software development lifecycle constitute of multiple phases, it’s recommended as per the case study results to design the productivity metric for each phase to effectively measure the productivity level of each phase separately because of the different natures of inputs/outputs of each phase. For example number of SLOC is output from development phase where number of use case is output from the design phase, where both outputs can’t be measured together.

- Case study results proved the impact of the non-technical aspects/factors that affect the productivity measurements. This needs to be weighted/discounted from the effective total productivity based on productivity models designed as described above. While orgzniations need to take corrective actions to maximize the effective developers utilizations to improve the throughput towards their clients.
- Code reusability is one of the current important factors which is very common these days and contribute in improving organizations productivity figures if correctly utilized to satisfy the new requirements.
- Software manufacture mode is also one of the important factors that impact the productivity measurement. For example defect-fixing based project type will differ from the green-field project type where software product is developed from scratch. Designing the productivity measurement should consider the project type factor as well.

5- CONCLUSIONS

In software development literature, productivity is a complex concept that needs to be tackled depending on the software project factors. There are technical and non-technical factors which has a considerable effect on the software productivity. This paper sheds a light on the main factors both technical and non-technical that impact software productivity. It also explores the different best practices adopted by software engineers and shows its effect as seen by the industrial software engineers in different domains. List of recommendations and corrective actions has been provided as way for continual improvement. Finally, software productivity measurement is a learning activity and therefore, historical information related to factors and measures is required. Hence, in order to learn and keep improving software engineering processes, organizations may continuously record and accumulate diverse metrics of their project. However, establishing this record process is not enough; organizations should achieve a balance in the investment of recording the required data and its future in order to accomplish improved goals. In this direction, there have been some national and international research organizations responsible of the creation of specific projects for establishing data banks of productivity measures along with many measurement factors, but generally these projects have ended fading away. Therefore, the creation and promotion of new data banks, mainly international, will enable a solid start point to further research in this important area.

ACKNOWLEDGMENT

The authors wish to thank HP organization for supporting this research study through survey sessions with software development team in different technological domains and industrial fields.

REFERENCES

[1] A. J. Albrecht. Measuring application development productivity In Proc. Joint SHARE/GUIDE/IBM Application Development Symposium, pp. 83–92, 1979.

[2] F. C. Brodbeck and M. Frese, editors. Produktivit¨at und Qualit¨at in Software-Projekten. R. Oldenbourg Verlag, 1994.

[3] W. S. Humphrey and N. D. Singpurwalla, N.D. 1991. “Predicting (individual) software productivity,” *IEEE Trans. Software Engineering*, vol. 17, pp. 196 – 207, Feb.1991.

[4] J. D. Blackburn, G. D. Scudder, and L. N. Van Wassenhove. Improving speed and productivity of software development: A global survey of software developers. IEEE Transactions on Software, 1996.

[5] B. W. Boehm and P. N. Papaccio. Understanding and controlling software costs. IEEE Trans. Softw. Eng., 14(10), pp. 1462–1477, 1988.

[6] B. W. Boehm, C. Abts, A. W. Brown, S. Chulani, B. K Clark, E. Horowitz, R. Madachy, D. Reifer, and B. Steece Software Cost Estimation with COCOMO II. Prentice-Hall, 2000.

[7] B. Lakhanpal. Understanding the factors influencing the performance of software development groups: An exploratory group-level analysis. Inform. Software Tech., 35(8), pp. 468–473, 1993.

[8] F. C. Brodbeck. Software-Entwicklung: Ein T¨atigkeitsspektrum mit vielf¨altigen Kommunikationsund Lernanforderungen. In Brodbeck and Frese, pp. 13–34.

[9] C. Wohlin and M. Ahlgren. Soft factors and their impact on time to market. Software Qual. J., 4(3), pp.189–205, 1995.

[10] K. Maxwell, L. VanWassenhove, and S. Dutta. Software development productivity of European space, military and industrial applications. IEEE Trans. Softw. Eng., 22(10), pp. 706–718, 1996.

[11] P. D. Chatzoglou and L. A. Macaulay. The importance of human factors in planning the requirements capture stage of a project. Int. J. Proj. Manag., 15(1), pp. 39–53, 1997.

[12] Grönroos, C., & Ojasalo, K., Service productivity: Towards a conceptualization of the transformation of inputs into economic results in services. *Journal of Business Research, 57*(4), pp. 414-423, 2004.

[13] Koskinen, K. U., Boundary brokering as a promoting factor in competence sharing in a project work context. *International Journal of Project Organisation and Management, 1*(1), pp. 119-132, 2008.

[14] J. D. Blackburn, G. D. Scudder, and L. N. Van Wassenhove Improving speed and productivity of software development: A global survey of software developers. IEEE Transactions on Software Engineering, 22(12), 1996.

[15] C. Jones. Software Assessments, Benchmarks, and Best Practices. Addison-Wesley Information Technology Series. Addison-Wesley, 2000.

[16] D. Port and M. McArthur. A study of productivity and efficiency for object-oriented methods and languages. In Proc. Sixth Asia-Pacific Software Engineering Conference (APSEC ’99), IEEE Computer Society, pp. 128–135, 1999.

[17] C. Wohlin and M. Ahlgren. Soft factors and their impact on time to market. Software Qual. J., 4(3), pp.189–205, 1995.

[18] R. Berntsson-Svensson and A. Aurum. Successful software project and products: An empirical investigation. In Proc. 2006 ACM/IEEE International Symposium on Empirical Software Engineering (ISESE ’06), ACM Press, pp. 144–153, 2006.

[19] T. K. Abdel-Hamid. The dynamics of software project staffing: A system dynamics based simulation approach. IEEE Transactions on Software Engineering, 15(2), pp.109–119, 1989.

[20] R. P. Cerveny and D. A. Joseph. A study of the effects of three commonly used software engineering strategies on productivity software enhancement Information & Management, 14(5), pp.243–251, 1988.

[21] G. R. Finnie, G. E. Wittig, and D. I. Petkov. Prioritizing software development productivity factors using the analytic hierarchy process. Journal of Systems and Software, 22(2), pp. 129–139, 1993.

[22] Z. Jiang, P. Naud´e, and C. Comstock. An investigation on the variation of software development productivity. International Journal of Computer and Information Science and Engineering, 1(2), pp. 72–81, 2007.

[23] S. Alper, D. Tjosvold, and K. S. Law. Conflict management, efficacy, and performance in organizational teams. Personnel Psychology, 53, pp. 625–642, 2000.

[24] R. H. Rasch. An investigation of factors that impact behavioral outcomes of software engineers. In Proc. SIGCPR. ACM Press, 1991.

[25] J. Turcotte and L. W. Rennison. The link between technology use, human capital, productivity and wages: Firm-level evidence. International Productivity Monitor, 9, pp. 25–36, 2004.