CHAPTER 8

Factors That Influence Productivity: A Checklist

Stefan Wagner, University of Stuttgart, Germany

Emerson Murphy-Hill, Google, USA

Introduction

In all areas of professional work, there are a lot of factors that influence productivity. Especially in knowledge work, where we do not have easily and clearly measurable work products, it is difficult to capture these factors. Software development is a type of knowledge work that comes with even more specific difficulties, as software developers deal nowadays with incredibly large and complex systems.

Yet, developers have to run software projects, manage other software developers, and optimize software development to make projects more competitive. Hence, we need a good overview of factors influencing productivity in software development so that developers and managers know what to focus and work on. Developers and managers probably have learned some factors that affect individual productivity, as well as team productivity, from experience. Even more useful, however, would be a list of factors that empirically have been shown to impact productivity in a more general way.

We provide such a list in this chapter as a kind of checklist that a developer or software manager can use to improve productivity. We will discuss technical factors related to the product, the process, and the development environment, as well as soft factors related to the corporate culture, the team culture, individual skills and experiences, the work environment, and the individual project.

C. Sadowski and T. Zimmermann (eds.), *Rethinking Productivity in Software Engineering*,
https://doi.org/10.1007/978-1-4842-4221-6_8

A Brief History of Productivity Factors Research

There has been research on productivity in software development since the 1970s. The first studies have been very influential, and several of the factors we have compiled in this chapter were identified back then. However, some of the factors from the 1970s, such as chief programmer team usage or previous experience with operational computers, have become less important over time.

The 1980s saw a more systematic collection of data with, for example, a series of books by Jones [7]. But researchers also realized the importance of psychological and sociological factors. Most important, as De Marco and Lister discuss in Peopleware [3], are aspects such as employee turnover and the developers' workplace. They also emphasize product quality as an important factor for productivity. Around the same time, the most famous effort prediction model was published, COCOMO [6].

Maybe as a result of Peopleware, the 1990s saw more research on soft factors. There were studies on project duration and the usage of object-oriented approaches. In the 2000s, no completely new aspects were introduced, but the understanding of several factors, such as requirements volatility or customer participation, was investigated.

We will summarize the main factors from these decades of research and add a brief review of newer factors that have been investigated in the 2010's so far.

The List of Technical Factors

The following three tables show the product, process, and environment factors that have been found in the literature to have an impact on software development productivity. The factors in the tables are sorted alphabetically.

Product Factors

The list of product factors has seen little change over the past ten years. There are several factors related to size and complexity. Software size usually means the size of the code needed for the software system. Product complexity tries to capture how difficult it is to implement the system with more or less code. In any case, the extent and complexity of the software including its data is a major factor that reduces productivity. Related are also technical dependencies. Newer studies have focused on the dependencies between different software modules or components and how this is reflected in social dependencies in the development team. A high number of dependencies reduces productivity.

Factor	Description	Source
Developed for reusability	To what extent should the components be reusable?	[1]
Development flexibility	How strong are the constraints on the system?	[1]
Execution time constraints	How much of the available execution time is consumed?	[1]
Main storage constraint	How much of the available storage is consumed?	[1]
Precedentedness	How similar are the projects?	[1]
Product complexity	The complexity of the function and structure of the software.	[1]
Product quality	The quality of the product influences motivation and hence productivity.	[1]
Required software reliability	The level of reliability needed.	[1]
Reuse	The extent of reuse.	[1]
Software size	The amount of code in the system.	[1]
User interface	The degree of complexity of the user interface.	[1]
Technical dependencies	Data-related or functional dependencies such as call graphs or coupled changes.	[5, 11]

A further set of factors that are related are constraints on execution time, main storage constraints, and constraints overall, what we term development flexibility. This could be integrated into a single factor. However, the first two describe more specific real-time and embedded systems, while the latter can also cover other constraints. An example of these constraints might be the use of specific operating systems or database systems or a high number of concurrent users. Additional constraints potentially slow down development.

Furthermore, the requirements on the user interface play an important role. It is a difference if a graphical user interface has to be developed or if the product is a background service. Sophisticated user interfaces typically reduce productivity.

The next product factors are related to quality. The current product quality makes it easier or more complicated to work on the software. Higher requirements on reliability and reusability can increase the effort needed. New publications widen this also to other quality attributes.

Finally, what the organization has done before plays a role: precedentedness describes how similar the project in question is to existing software, and reuse describes how much of the new software can be achieved by reusing existing software (e.g., internal or open source).

Process Factors

The next category of factors are still technical but relate more to the process than the product itself. These factors are related to the project: project length and project type. Longer projects are more difficult to organize but benefit more from rules and custom tools. A more recent study [8] distinguished between development and integration projects. Development projects create most of the software during the project, while integration projects mostly connect and configure existing software. They found that integration projects are more productive.

Factor	Description	Source
Agile	Is an agile development process used?	[10, 12, 13]
Architecture risk resolution	How are the risks mitigated by architecture?	[1]
Completeness of design	The amount of the design that is completed when coding starts.	[1]
Early prototyping	Early in the process prototypes are built.	[1]
Effective and efficient V&V	The degree to which defects are found and the required effort therein.	[1]
Hardware concurrent development	Is the hardware developed concurrently?	[1]
Outsourcing and global distribution	Degree of outsourcing of the work of the project.	[9]
Platform volatility	Time span between major changes.	[1]
Process maturity	The well-definedness of the process.	[1]
Project duration	Length of the project.	[1]
Project type	Integration or development project.	[8]

From the next factors, we see that different development activities have an impact on productivity. Architecture risk resolution is important in architecture design and evolution. The completeness of design before the start of coding impacts how much changes need to be done later. Finally, effective and efficient V&V (verification & validation) describes suitable tests, reviews, and automated analysis. Early prototyping can increase productivity because requirements can be clarified and risks can be resolved. Today, this is often replaced by iterative and incremental development. Such a development probably is able to better deal with volatile requirements, but the completeness of the design during initial coding is low.

Most systems today are not completely stand-alone but rely on specific platforms or hardware. If the platform changes frequently (platform volatility), it creates a lot of adaptation effort. The concurrent development of hardware also means that it is difficult to rely on the hardware and might require adaptation efforts in the software.

The last factors are about the process model and the distribution of the work. A general factor is the process maturity, meaning how well-defined the development process is. In the recent years, research has focused on agile processes and found that they impact productivity. A further aspect of recent studies is outsourcing and global distribution of the project.

Development Environment

In the last category, we group factors that are not part of the product but not directly part of the process either.

Factor	Description	Source
Documentation match to life-cycle needs	How well the documentation fits the needs	[1]
Domain	Application domain such as embedded software, management information system, or web application	[4]
Programming language	The programming language used	[1, 21]
Use of software tools	The degree of tool use	[1]
Use of modern development practices	For example, continuous integration, automated testing, or configuration management	[1]

A very general factor is the domain of the application to be developed. Embedded software systems, for example, often have specific aspects such as cross-compiling that make development more difficult. Also quite general is the programming language used and the use of modern development practices. The latter includes methods such as continuous integration or automated tests that often come with agile development processes but are not restricted to them. Furthermore, the use of software tools such as modern IDEs or test frameworks impacts productivity. Finally, we also count the match of documentation to environmental factors. In particular, it is important if the documentation fits the needs of the current state of development.

The List of Soft Factors

As most people in a software engineering team have a technical background, we tend to focus on technical aspects. Yet, especially for productivity, many more soft factors play an important role. We will discuss the soft factors we have found in the following five categories: Corporate Culture contains the factors that are on a more company-wide level, whereas Team Culture denotes similar factors on the team level. In Individual Skills and Experiences, we summarize factors that are related to individuals. Work Environment stands for properties of the environment such as the workplace itself. Finally, project-specific factors are in the Project category. We sort the factors in each category again alphabetically.

Corporate Culture

We start with the factors related to the culture of the complete organization. All these factors could also be interesting on the team level, but the culture of a company overall reflects down to the teams as well. Researchers have studied the three factors credibility, fairness, and respect especially on the organizational level.

Factor	**Description**	**Source**
Credibility	Open communication and competent organization	[1]
Fairness	Fairness in compensation and diversity	[1]
Respect	Opportunities and responsibilities	[1]

Credibility is probably the most general factor that describes that communication is open overall in the company and the organization is competent in what it is doing. In our context, this could mean, for example, that there is an understanding on the organizational level of how to plan and run software projects. In fairness, we include equal payment opportunities for all employees and diversity in terms of gender or background in the organization. Respect, finally, means that the organization sees their employees not only as "human resources" but as people; management gives the employees opportunities and trusts them with responsibilities.

Team Culture

There has been considerably more research on the team level than on the corporate level. There can be strong differences between teams in the same company. The higher number of studies brought us eight factors in team culture influencing productivity.

Factor	**Description**	**Source**
Camaraderie	Social and friendly atmosphere.	[1]
Clear goals	How clearly defined are the group goals?	[1]
Communication	The degree and efficiency of which information flows in the team.	[1]
Psychological safety	The atmosphere is safe for risk-taking.	[14, 15]
Sense of eliteness	The feeling in the team that they are superior.	[1]
Support for innovation	To what degree assistance for new ideas is available.	[1]
Team cohesion	The cooperativeness of the stakeholders.	[1]
Team identity	A common identity of the team members.	[1]
Turnover	The amount of change in the personnel.	[1]

Camaraderie means a social and friendly atmosphere where team members socialize but also help each other. The second factor in this category consists of clear goals that are necessary so that all team members work toward the same objective. Most general is the factor communication that includes the degree as well as the efficiency of information flow inside the team. In general, what is surprising in the studies is that communication effort is positive for productivity. In discussions, we often hear that communication should be reduced to decrease unnecessary work. However, the actual problems seems to be the increase of communication effort when putting more and more people on a project. Yet, a high fraction of effort on communication seems like a good investment.

Psychological safety is similar to camaraderie but more specifically refers to an atmosphere where individual developers can take risks and share personal information, but know that teammates will handle these risks with respect and kindness. This is a factor that more recently came into productivity discussions in the context of software projects because of a large study at Google [14]. Also similar but aiming in a different direction is the sense of eliteness of the team. If the team believes that they are the best engineers always building the highest-quality software, they are more likely to go the extra mile to actually achieve this.

Also related to psychological safety is support for innovation. This contains to some degree safety for taking risks, but it also means that the team members are open to bring in innovations and also change the way they work. Yet another view on this is team cohesion. Team cohesion describes how well all team members are willing to work together. This does not necessarily include a social and friendly atmosphere but a professional approach to working together.

A common team identity also seems to support productivity, probably by influencing other factors such as camaraderie or the sense of eliteness. Finally, the turnover in the team might be influenced by the factors mentioned so far. Team changes could also be ordered by management because of other influences. In any case, less turnover is better for productivity, and it is one of the few factors that we can easily measure.

Individual Skills and Experiences

Besides teams, individual skills and experiences are the most well-studied. We found it notable that although experience is often brought up and is in interviews considered important, in empirical studies it is rather insignificant. By far more interesting is the capability of the developers. Hence, this suggests that being in a profession for a long time does not necessarily make one productive.

Factor	**Description**	**Source**
Analyst capability	The skills of the system analyst	[1]
Application domain experience	The familiarity with the application domain	[1]
Developer personality	Individual personality and the mix of different personalities on the team	[1, 19]
Developer happiness	Positive experiences leading to positive emotions	[16–18]
Language and tool experience	The familiarity with the programming language and tools	
Manager application domain experience	The familiarity of the manager with the application	[1]
Manager capability	The control of the manager over the project.	[1]
Platform experience	The familiarity with the hardware and software platforms	[1]
Programmer capability	The skills of the programmer	[1]

Therefore, we have factors for the analyst capability, the manager capability, and the programmer capability. Each refers to the skills of the individuals in their respective roles. For each role, these skill sets will differ, but there is thus far no fixed set of skills necessary for the roles that came out of the studies.

Experience does play a role but more in the sense of the experience with application domains and platforms. We have the three factors of application domain experience, manager application domain experience, and platform experience. The first two refer to how long and with what intensity the developers and managers have worked on software in a specific application domain. The latter refers to the experience of the individuals with a hardware and/or software platform such as the iOS operating system for mobile Apple devices.

Developer personality has been investigated in many empirical studies. Few measure personality according to the state of the art in personality psychology. A more recent study [19] found only one personality trait—conscientiousness—impacted productivity (positively).

Similarly to the study of personalities, another important psychological area has recently been investigated: the emotions of developers. Several studies [16–18] looked at the relationship of happiness of developers and their productivity. They found indeed that happy developers are more productive. You can find more details in Chapter 10.

Work Environment

This category of factors could be seen on the organizational or team level. Yet, as there are five factors, we decided to put them in their own category. They describe the direct work environment of the software engineers.

Factor	Description	Source
E-factor	This environmental factor describes the ratio of uninterrupted hours and body-present hours.	[1]
Office layout	Private or open-plan office layout.	[22]
Physical separation	The team members are distributed over the building or multiple sites.	[1]
Proper workplace	The suitability of the workplace to do creative work.	[1]
Time fragmentation	The amount of necessary "context switches" of a person.	[1]
Telecommunication facilities	Support for work at home, virtual teams, video conferencing with clients.	[1]

The e-factor introduced by DeMarco and Lister in Peopleware [3] emphasizes that uninterrupted time for work is important for productivity. Chapter 9 discusses this in more detail, and Chapter 23 shows an idea to improve the e-factor.

Although we have not found studies focusing specifically on software engineering teams, there are several studies on office layout that should apply in our context. In software companies, we frequently see open-plan offices with the reasoning that interaction between team members is important. A recent large study [22] found no evidence that this is actually the case. Instead, interruptions are much higher; hence, the e-factor becomes worse in open-plan offices.

Distributed development of software, meaning software teams physically distributed over several locations in potentially several different time zones, is common today. There is a considerable body of work on the potential problems with this working mode. It can have a negative effect on productivity.

Also, the workplace itself has an effect on productivity. There are studies investigating aspects such as if there are windows and natural light or the size of the room and space on a desk. Time fragmentation is related to the e-factor but covers more the aspect of how many different projects and kinds of tasks you have to work on. This results in costly context switches that could be avoided if you could focus on a single project.

Finally, proper telecommunication facilities are important so that you can work from home, work efficiently part-time, or interact efficiently with other team members who are in another physical location.

Project

Finally, there are factors related to the individual project that are not technical in the sense that they come from the technology or programming language. Instead, the people associated with the project influence them.

Factor	Description	Source
Average team size	Number of people on the team	[1]
Requirements stability	The number of requirements changes	[1, 4, 20]
Schedule	The appropriateness of the schedule for the development task	[1]

There are many studies looking into the relationship of team size and productivity. It is well established that larger teams lead to exponentially increasing communication efforts that, in turn, lead to lower productivity. Newer, agile software development processes therefore often recommend team sizes of about seven.

Also, the requirements stability over a project has been the subject of several studies. Highly unstable requirements lead to time, effort, and budget overruns; overall demotivation; decreased efficiency; and the need for post-implementation [20]. Again, agile development processes focus on this problem by reducing development cycles to a few weeks.

Finally, the planned project schedule needs to fit the actual work to be done. Several studies show that schedules that are too tight in effect reduce the productivity.

Summary

Our taxonomy of factors influencing software development productivity is extremely diverse. The technical factors range from detailed product factors, such as execution time constraints, to general environment factors such as the use of software tools. The soft factors have been investigated on the corporate, team, project, and individual levels. For specific contexts, it will be necessary for practitioners to look into each of these factors in more detail. We hope that this chapter can be used as a starting point and checklist for productivity improvement in practice.

Key Ideas

These are the key ideas from this chapter:

- The major factors influencing software development productivity can be summarized in a checklist for developers and managers.
- Some of the relevant research on productivity factors is decades old.

Acknowledgments

We are grateful to Melanie Ruhe for previous discussions on productivity and productivity factors.

Appendix: Review Design

This chapter is not meant to be a full-fledged academic literature review. Instead, we used our prior literature review [1] as a start and updated it with a search on Google Scholar. For the analysis, we also reused the search string from [1] to stay consistent: software AND (productivity OR "development efficiency" OR "development effectiveness" OR "development performance")

In contrast to the old review, however, we looked at only the first 30 results from 2017 to 2018 in Google Scholar. Of those results, we extracted any new relevant productivity factors from empirical studies. We did not use studies that only validated factors already on the list to keep this article concise. We also noted that while most of the factors come from academic papers investigating these factors in more detail, the old literature review [1] also included the books by Boehm [6] and Jones [7] as a baseline. They do not investigate single factors but use a set of factors to discuss productivity.

Finally, the extracted academic studies have limitations, such as some of them use lines of code per person-hour as a productivity measure. This is easy to measure but has significant problems because more code is not necessarily good. In many instances, less code is actually better as long as it fulfils the customer's requirements and needs. We decided to not exclude these studies, however, as the identified factors still might be interesting.

References

[1] Wagner, Stefan and Ruhe, Melanie. "A Systematic Review of Productivity Factors in Software Development." In Proc. 2nd International Workshop on Software Productivity Analysis and Cost Estimation (SPACE 2008). Technical Report ISCAS-SKLCS-08-08, State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, 2008.

[2] Hernandez-Lopez, Adrian, Ricardo Colomo-Palacios, and Angel Garcia-Crespo. "Software engineering job productivity—a systematic review." International Journal of Software Engineering and Knowledge Engineering 23.03 (2013):387–406.

[3] T. DeMarco, T. Lister. "Peopleware. Productive Projects and Teams." Dorset House Publishing, 1987.

[4] Trendowicz, Adam, Münch, Jürgen. "Factors Influencing Software Development Productivity – State of the Art and Industrial Experiences." Advances in Computers, vol 77, pp. 185–241, 2009.

[5] Cataldo, Marcelo, James D. Herbsleb, and Kathleen M. Carley. "Socio-technical congruence: a framework for assessing the impact of technical and work dependencies on software development productivity." Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement. ACM, 2008.

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

[8] Lagerström, R., von Würtemberg, L.M., Holm, H., Luczak, O. Identifying factors affecting software development cost and productivity. Software Qual J (2012) 20: 395. https://doi.org/10.1007/s11219-011-9137-8.

[9] Tsunoda, M., Monden, A., Yadohisa, H. et al. Inf Technol Manag (2009) 10: 193. https://doi.org/10.1007/s10799-009-0050-9.

[10] Kautz, Karlheinz, Thomas Heide Johanson, and Andreas Uldahl. "The perceived impact of the agile development and project management method scrum on information systems and software development productivity." Australasian Journal of Information Systems 18.3 (2014).

[11] Cataldo, Marcelo, and James D. Herbsleb. "Coordination breakdowns and their impact on development productivity and software failures." IEEE Transactions on Software Engineering 39.3 (2013): 343–360.

[12] Cardozo, Elisa SF, et al. "SCRUM and Productivity in Software Projects: A Systematic Literature Review." EASE. 2010.

[13] Tan, Thomas, et al. "Productivity trends in incremental and iterative software development." Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement. IEEE Computer Society, 2009.

[14] Duhigg, Charles. "What Google learned from its quest to build the perfect team." The New York Times Magazine 26 (2016): 2016.

[15] Lemberg, Per, Feldt, Robert. "Psychological Safety and Norm Clarity in Software Engineering Teams." Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering. ACM, 2018.

[16] Graziotin, D., Wang, X., and Abrahamsson, P. 2015. Do feelings matter? On the correlation of affects and the self-assessed productivity in software engineering. Journal of Software: Evolution and Process. 27, 7, 467–487. DOI=10.1002/smr.1673. Available: https://arxiv.org/abs/1408.1293.

[17] Graziotin, D., Wang, X., and Abrahamsson, P. 2015. How do you feel, developer? An explanatory theory of the impact of affects on programming performance. PeerJ Computer Science. 1, e18. DOI=10.7717/peerj-cs.18. Available: https://doi.org/10.7717/peerj-cs.18.

[18] Graziotin, D., Fagerholm, F., Wang, X., & Abrahamsson, P. (2018). What happens when software developers are (un)happy. Journal of Systems and Software, 140, 32-47. doi:10.1016/j.jss.2018.02.041. Available: https://doi.org/10.1016/j.jss.2018.02.041.

[19] Zahra Karimi, Ahmad Baraani-Dastjerdi, Nasser Ghasem-Aghaee, Stefan Wagner, Links between the personalities, styles and performance in computer programming, Journal of Systems and Software, Volume 111, 2016, Pages 228–241, https://doi.org/10.1016/j.jss.2015.09.011.

[20] D. Méndez Fernández, S. Wagner, M. Kalinowski, M. Felderer, P. Mafra, A. Vetrò, T. Conte, M.-T. Christiansson, D. Greer, C. Lassenius, T. Männistö, M. Nayabi, M. Oivo, B. Penzenstadler, D. Pfahl, R. Prikladnicki, G. Ruhe, A. Schekelmann, S. Sen, R. Spinola, A. Tuzcu, J. L. de la Vara, R. Wieringa, Naming the pain in requirements engineering: Contemporary problems, causes, and effects in practice, Empirical Software Engineering 22(5):2298–2338, 2017.

[21] Ray, B., Posnett, D., Filkov, V., & Devanbu, P. (2014, November). A large scale study of programming languages and code quality in github. In Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (pp. 155–165). ACM.

[22] Jungst Kim, Richard de Dear, "Workspace satisfaction: The privacy-communication trade-off in open-plan offices," Journal of Environmental Psychology 36:18–26, 2013.