Spring 6-10-2017

SOFTWARE PROGRAMMER PRODUCTIVITY: A COMPLEMENTARY-BASED RESEARCH MODE

Natallia Pashkevich
Linnaeus University, natallia.pashkevich@lnu.se

Darek M. Haftor
Linnaeus University, Växjö, darek.haftor@lnu.se

SOFTWARE PROGRAMMER PRODUCTIVITY: A COMPLEMENTARY-BASED RESEARCH MODEL

Research in Progress

Pashkevich, Natallia, Linnaeus University, Växjö, Sweden, natallia.pashkevich@lnu.se

Haftor, Darek, M., Linnaeus University, Växjö, Sweden, darek.haftor@lnu.se

Abstract

The identification of the factors that condition a software programmer's productivity remains a key challenge for both scholars and practitioners. While a number of studies have focused on the impact of one or a few particular factors, the way these factors jointly condition programmer productivity is still unknown. This paper presents a conceptual model aimed at a comprehensive understanding of the factors that complement each other to govern the productivity of a software programmer. The model is based on complementarity theory and its systems approach and addresses an individual worker's productivity, which accounts for cognitive, technological, and organizational characteristics. The analyzed factors are organized into a system of complementarities, offering two propositions that specify the conditions of a programmer's productivity. The model's key contribution lies in its unique configuration of two systems of complementarities, which have the potential to add to the literature on the productivity of software programmers. The proposed model can be employed as a guidance for the design of empirical investigations of the conditions of individual software programmers' productivity as well as information worker productivity in general.

Keywords: Complementarity, Individual productivity, Software programming, Systems approach.

1 Introduction

The notion of software programming, as a central activity of a system development, constitutes one of the oldest research domains in the Information Systems literature (Kirikova et al., 2012). The determinants of software programmers' productivity have always been a concern for researchers (Petersen, 2011). Several studies report the importance of different factors, including IT tools, work tasks, and personal and organizational characteristics that condition the productivity of software programmers (de Barros Sampaio et al., 2010). However, addressing these factors as independent from each other does not provide a comprehensive understanding of productivity. The recent literature demonstrates that, to obtain a productivity gain from an information worker, it is necessary to focus on how the conditioning factors complement each other and jointly govern the productivity (Brynjolfsson and Milgrom, 2013). The literature, however, does not identify the factors that need to be synchronized to increase individual programmer productivity. Therefore, the objective of this paper is to develop a model that specifically identifies the factors that, when synchronized together, condition the productivity of an individual software programmer.

In this paper, we propose a model to guide empirical research on individual programmer productivity growth, where cognitive, technological, and organizational factors are synchronized into a system of mutually interacting factors. All included factors are obtained from existing independent empirical studies (Ahearne et al., 2005; Amabile, 1996; Baer et al., 2003; Benbasat and Dexter, 1985; Hayes and Allinson, 1997; Kirton, 1976; 1994; MacCormack et al., 2001; Riding and Douglas, 1993). The uniqueness of the proposed model lies in the specific combination of these elements into a system of complementary factors. The aim of this study is to enable a better understanding of how the individual

productivity of software programmers can be improved and provide clear strategies for the management of individual productivity.

The remainder of this paper is organized as follows. Section 2 revisits the literature on the productivity of software programmer and information worker productivity. Section 3 provides the research model formulation. Section 4 summarizes the propositions that describe a software programmer's productivity. Section 5 ends the paper with a discussion of the potential contributions of this model and its limitations.

2 Literature Review

Previous studies show that the programmer is a particular instance of a broader category of information worker (Lal, 2005), sometimes referred to as knowledge worker (Davenport, 2005) or white-collar worker (Kelly, 2008). An essential characteristic of information workers is that they primarily receive and process information and, then, generate information as an output, as compared to the physical worker's processing matter (Wolff, 2005). Some studies show that, while the various conditions of information worker productivity versus physical workers may differ (Drucker, 1999), they tend to be homogeneous within the two categories (North and Gueldenberg, 2011). Therefore, we review both studies on information worker productivity and on the productivity of software programmers, at the individual level.

Peter Drucker, a key advocate of the notion of information workers and their inherent peculiar character, proposed that the fundamental determinants of information worker productivity include task definition, degree of self-management or autonomy, continuous innovation, learning and skill development as part of the work, and focus on quality rather than quantity (Drucker, 1999). Independent information worker studies show that central determinants of information worker productivity include cognitive styles of workers (Agarwal and Prasad, 1999; Zmud, 1979), motivation (Frey and Osterloh, 2002), training for skills (Kessels, 2001; Kraiger, 2003), decision-making structures or degree of autonomy (Kozlowski and Bell, 2003; Schneider and Smith, 2004), type of work processes (Parker et al., 2001; Sokoya, 2000), including multi-tasking (Appelbaum et al., 2008; Bell et al., 2005) and technology support, particularly the use of IT systems (Aral et al., 2012; Jain and Kanungo, 2005).

Some studies on software programmer's productivity demonstrate that the productivity of a software programmer may be conditioned by an array of factors, including the work processes utilized (Banker et al., 1991; Cockburn and Highsmith, 2001; Jiang et al., 2007), the project management practices adopted (Jones, 2000; Sharp et al., 2009), the IT tools employed (Banker, 1987; Clincy, 2003), the programming languages utilized (Jiang and Naude, 2007; McConnell and Complete, 1993), the quality of requirement specifications provided (Lokan, 2001; Scacchi and Hurley, 1995), the programmer's characteristics (Capretz, 2003; Cougar and Zawacki, 1980; Darcy and Ma, 2005), and various organizational factors, such as decision-making (Tessem, 2011; Toffolon and Dakhli, 2007) and organizational culture (Donaldson and Siegel, 2001; Wagner and Ruhe, 2008). A key limitation of this vast body of research is that each study investigates a particular factor in isolation (Ondrej et al., 2012); also, these studies typically are context dependent (de Barros Sampaio et al., 2010).

Over the last decades, a specific theory of complementarities has been advanced, and its most elaborated notion is provided by the recent developments in organizational economics (Roberts, 2007). This offers a specific conception of how various organizational practices may fit each other and which patterns of fit may exist to produce a desired outcome (Brynjolfsson and Milgrom, 2013). This theory argues that positive complementarities emerge when the marginal return to one activity or resource increases in the presence of another activity or resource (Milgrom and Roberts, 1995). Complementarity theory emerged from empirical observations, which show that changing only one or two operational factor(s) at a time may not come close to all benefits that are available when a system of specific complementarities is addressed in a purposeful and synchronized manner.

Complementarity theory is a kind of meta-theory; it does not focus on the identification of the factors

that should be accounted for, but addresses how the chosen factors should be accounted for. Furthermore, complementarity theory offers two different approaches to investigation (Ennen and Richter, 2010). The interaction approach studies the interaction of a few factors, two or three, and produces a detailed account of the nature of such an interaction. The interaction approach, however, disregards other factors that condition the studied phenomenon. The systems approach, on the other hand, focuses on all key factors simultaneously to account for the total "systemicity" of the investigated phenomenon, yet ignoring the specific nature of the local interactions between two or three factors (ibid.). Previous studies show diversity across the individual determinants of the productivity of a software programmer, each explaining some productivity variance. To the best of the authors' knowledge, there are no current attempts to offer a more comprehensive understanding of the productivity of a programmer. Therefore, the research model proposed in this study assumes the systems approach of complementarity theory to investigate what factors need to be synchronized together in a complementary manner, thereby conditioning the software programmer's productivity.

3 Research Model of Software Programmer Productivity

The proposed research model for the productivity of a software programmer is outlined in Figure 1. The formulation process of the model included a careful review of all relevant studies; hence, no new factors were developed in this paper. The literature review provided both a list of potential factors to be included in the model and an account of which factors tend to dominate the productivity of information workers and software programmers. The most central factor is the *cognitive style* of the programmer (detailed further). Therefore, the subsequent formulation of the model explored the degree of fit or complementarity between both the available factors and their fit with the cognitive style factor. This process resulted in a model with eight factors, besides the dependent factor. While these elements have a certain interaction with each other, they are listed here in a linear manner by necessity.

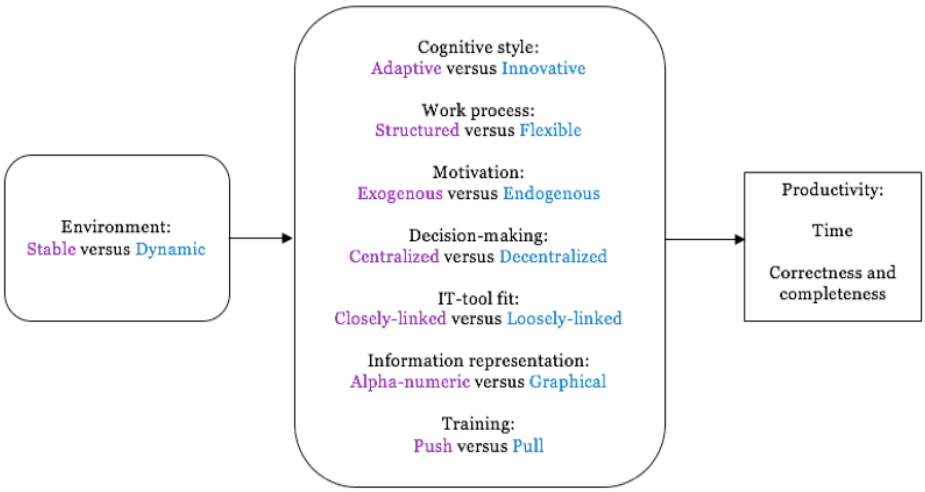

Figure 1. Research model for the productivity of an individual software programmer

Integrating the empirical evidence obtained from prior studies, we developed two theoretical propositions regarding complementarities that, when matched correctly, can affect individual programmer's positively. The formulated propositions, along with a more detailed support for their development, are specified below.

3.1 Cognitive style

The essential component of the productivity of a software programmer is the programmer, a human professional who receives some requirement specification and proceeds with the production of a piece of software. As noticed, this context represents the category of information workers where a key act of

information processing is conducted through human cognitive faculties (Pyöria, 2005) rather than physical capabilities. This means that human cognition is a central factor for the comprehension of information work processes. Studies in psychology have shown that humans may be described in terms of various kinds of cognitive characteristics, which strongly influence the cognitive acts and information processing ability (Agor, 1984; Rowe and Mason, 1987). While there are several partly competing and partly overlapping categorizations of human cognition, in this paper we adopt the Kirton scheme of adoption–innovation (Kirton, 1976; 1994). This scheme argues that human cognition may be characterized along a spectrum, from mainly adaptive behavior to mainly innovative behavior. This categorization reflects differences in human information processing, problem-solving procedures, decision-making modes, and the need for autonomy as well as responses to changes, among others (ibid.). The adaptive cognitive character tends to operate in a systematic and well-structured fashion, step-by-step, and prefers explicit and stable situations that are predictable. In contrast, the innovative cognitive character tends to operate in a creative and ad hoc manner and prefers a lack of imposed and pre-defined structures. We adopted this scheme of adaptive-innovative cognitive style for several reasons. One is that some studies show that both cognitive styles may perform well, or otherwise, depending on the context and its conditions (Kirton, 1994). Second, this scheme produces a good fit with the remaining factors included in the proposed research model. Third, the theory of the firm (Roberts, 2007) shows that a firm's success is conditioned by a deliberate coordination of two kinds of personalities: systematic and creative (Jablokow and Booth, 2006). Finally, the Kirton scheme is the most explored, elaborated, and well-tested categorization of human cognitive styles, offering well-developed measurement instruments (Brown, 2001). This paper argues that cognitive style constitutes a fundamental determinant of the productivity of a software programmer.

3.2 Work process and environment

A professional software programmer does not operate in isolation. The programmer executes a certain kind of work tasks that constitute a work process (Kock and McQueen, 1996). Some studies show that the character of a work process influences the performance of an information worker (Amabile, 1996; Keen and Scott-Morton, 1978). While several conceptions of work process categorization have been proposed (Weber and Wild, 2005), in this paper we assume the distinction between a structured work process versus a more flexible work process. This distinction shows a clear impact on the performance outcome (MacCormack et al., 2001). A more structured work process is explicitly pre-defined in terms of its constituting activities and their order; also, the inputs and outputs are typically carefully specified (Abdolmohammadi and Wright, 1987). Given this definition, we argue that a structured work process should be conducted by an adoptive cognitive style programmer, while a flexible work process should be carried on by an innovative cognitive style programmer. These factors should complement each other in relation to the productivity achieved in a manner that any alternative pairing of the two factors may not produce.

A software programmer receives a certain kind of input from its environment, the so-called requirement specification, that informs the programmer of which functions and information the software system is to offer (Bourque and Fairley, 2014). The quality of the input delivered from the programmer's environment may vary, depending on the context (MacCormack et al., 2001). To account for such variations and their impact on the programmer's performance, a distinction is adopted in this paper between stable and dynamic environments, in line with the contingency school thinking (Mintzberg, 1979). A stable environment means that the requirement specification received by the programmer is complete and will change little or nothing during the programming process (MacCormack et al., 2001). A dynamic environment implies that the requirement specification is not complete and changes, sometimes frequently, during the programming process (ibid.).

Given this definition, and with the support of the contingency school's conception of organizational performance (Burns and Stalker, 1961), we propose that a structured work process, as executed by a software programmer with an adaptive cognitive style, should be matched with a stable environment,

which provides the programmer with requirement specifications that are rather complete and subject to few or no changes during the programming process. On the other hand, the flexible work process, as executed by a programmer with an innovative cognitive style, should be matched with a dynamic environment, which provides the programmer with requirement specifications that are less complete and subject to more frequent changes during the programming process. It is assumed here that these combinations will generate positive complementarities in relation to a programmer's performance in a manner that any alternative matching of these factors may not achieve.

3.3 Motivation and decision-making

One of the most well researched and established factors that conditions a worker's productivity is motivation (Frey and Osterloh, 2002). This cognitive factor explains a significant portion of the worker productivity (Kasof et al., 2007) and is also central to a firm's performance (Roberts, 2007). Studies in psychology established a fundamental difference between exogenously and endogenously induced worker motivation (Amabile, 1996). These studies also show that the high performance of an adaptive cognitive style is matched with a greater degree of exogenous motivation, such as various compensation schemes, while high performance of an innovative cognitive style shows a greater fit with endogenously induced motivation, such as positive appreciation from a manager (Amabile et al., 1994; Baer et al., 2003). Therefore, we argue that the proposed elaborated model for the productivity of a software programmer includes exogenous motivation as a complement to the adaptive cognitive style and endogenous motivation complements the innovative cognitive style.

Another central area of information worker productivity is associated with worker freedom, autonomy, or self-management (Zabojnik, 2002). Operationally, this may be translated into decision-making freedom, authority, or power (Kozlowski and Bell, 2003). Decision-making, a cognitive act, is central for many information jobs both in terms of decision frequency and scope (Hunt et al., 1989; Rowe and Boulgarides, 1992), and has shown a significant impact on information worker productivity (Ahearne et al., 2005; Baer et al., 2003). Various studies in psychology demonstrate the presence of complementarity between the decision-making authority, in terms of centralization versus decentralization, and an individual's cognitive style (Amabile, 1996; Axtell et al., 2000; Oldham and Cummings, 1996). A higher degree of centralization provides a software programmer with fewer decisions and more given guidelines when a requirement specification is ambiguous, and should, therefore, constitute a complement to an adaptive cognitive style (Baer et al., 2003). On the other hand, a higher degree of decentralization offers a software programmer more freedom to decide on how to act and change actions, without the need to consult a manger or other decision maker, and should constitute a complement to an innovative cognitive style (Axtell et al., 2000; Oldham and Cummings, 1996).

3.4 Information technology tools

Working tools, such as various information and communication technologies (IT), are for an information worker what mechanical devices are for a physical worker (Zuboff, 1988). In the case of a software programmer, different IT systems may be utilized to produce a piece of software (Bruckhaus et al., 1996). Some studies argue that various IT systems may show different degrees of support, or fit, with various kinds of work processes (Collins, 2003; Mitchell and Zmud, 1999) and argue that the greater the degree of fit between the functionality of an IT system and the supporter work process, the higher the productivity (Das and Narasimhan, 2001). Some IT systems offer functionality with a high degree of customization to support a unique work process, so that specific IT system functions are directly linked to certain activities in a pre-defined work process (Beschab and Piot, 2007); we call such IT systems "closely-linked." Other IT systems offer more general functionality, which can support several versions of a certain kind of a work process, which, in turn, gives greater flexibility for various workarounds (Weber et al., 2008); we call such IT systems here "loosely-linked." We argue that the high performance of a software programmer with an adaptive cognitive style, who utilizes a structured work process, is complementary with an IT system that is closely-linked. On the other hand,

the programmer with an innovative cognitive style, who uses a flexible work process, is complemented by an IT system that is loosely-linked.

While the proposed complementarity of the IT system specifies a fit with the work process, we also suggest a fit between the IT system and the cognitive style of the software programmer. Some studies show that various cognitive styles have a preference with regard to the representational mode of information (Benbasat and Dexter, 1985; Riding and Douglas, 1993), a cognitive feature, with a key distinction between graphical representations versus alphanumeric representation modes (Yazici and Kluczny, 1993), which also gives rise to differences in information worker productivity (ibid.). Therefore, we argue that the high productivity of a software programmer with an adaptive cognitive style is complemented by the use of a predominately alphanumeric representation mode provided by the IT system utilized to support the work. On the other hand, the high productivity of a software programmer with an innovative cognitive style is complementary to the use of predominately graphical representational mode, provided by the IT system, as employed in the work process.

3.5 Training

Information workers, and software programmers in particular, are dependent on the skillful use of advanced work technologies, such as work processes and IT systems (Webster, 2012). The mastery of such work technologies influences work productivity (Hitt and Brynjolfsson, 1997) and may be supported and enacted with various training and education forms, which induce learning and develop work capabilities (Hayes and Allinson, 1997). Many studies have explored and established the significance of training and education of an information worker, which advances the cognitive ability to perform both new tasks and current tasks in a better manner (Berings et al., 2005; Hayes and Allinson, 1997; Sadler-Smith and Smith, 2004). Further, the cognitive character of information workers shows a correspondence to the training form (Amabile, 1996). In our attempt to account for this complementarity in the case of a software programmer, we argue that a programmer's training can assume two modes. The first mode stipulates training that is received mandatorily and in a pre-defined, well-structured fashion, prior to the initiation of the use of work technologies, such as the work process and the IT system; we can call this form "push-mode training." The second form of training is received optionally, on demand, both prior the initiation of the work technology use and after, with the desired frequency and when the programmer wishes to have it; we call this mode "pull-mode training." Given this definition, we argue that the high productivity performance of a software programmer with an adaptive cognitive style is complemented with push-mode training, while the high productivity of a software programmer who has an innovative cognitive style is complemented by pull-mode training.

3.6 Productivity of software programmer

In software engineering, a quantitative dimension is often characterized by a change in the quantity of a developed product for a given period, while a qualitative dimension refers to the quality of the software (Duncan, 1988). The input effort is usually a sum of the resources used to produce the output. In software programming, the main share of resources is the programmer's working hours spent on the software product development (Canfora et al., 2007). Since output measures, such as the physical length of the written codes, do not indicate whether an individual is productive or not, as these metrics depend on the coding style of the software programmer (Tomaszewski and Lundberg, 2005), the time devoted by the programmer to complete the assignment is assumed in this paper to characterize the quantitative dimension of the software programmer's productivity metric.

Software programming must also meet certain specific qualitative requirements, typically expressed in the input as the requirement specification (MacCormack et al., 2001). Indeed, variations in the quality of software produced may explain variations in the time needed to produce it. While software quality may be conceived in several terms, the notion of software quality adopted in this paper accounts for two critical dimensions: firstly, the degree of completeness of the software versus the pre-specified characteristics, as articulated by the requirements; secondly, the correctness of the software produced,

which refers to the number of defects in the produced software (Edberg and Bowman, 1996). Software programmer productivity is therefore understood in this paper as a function of the time devoted to produce the software and the software's degree of completeness and correctness.

4 Total System of Factors and its Propositions

The above-detailed factors are organized into a system of complementarities, offering two propositions that specify the conditions of a programmer's productivity:

Proposition 1: Maximal productivity is achieved by a software programmer with an *adaptive* cognitive style when a *structured* work process is employed in a *stable* environment and the programmer is motivated *exogenously*, exposed to a high-degree of *centralized* decision-making, and using an IT system that is both *closely-linked* to the work process and predominately offers the *alphanumeric* representation mode of information, and where work technology training is delivered in a *push-mode*, compared to any other configuration of these complementarities to the adaptive programmer.

Proposition 2: Maximal productivity is achieved by a software programmer with an *innovative* cognitive style when a *flexible* work process is employed in a *dynamic* environment, the programmer is motivated *endogenously*, exposed to a high-degree of *decentralized* decision-making, and using an IT system that is both *loosely-linked* to the work process and predominately offers *graphical* representation mode of information, and where work technology training is delivered in a *pull-mode*, compared to any other configuration of these complementarities to the innovative programmer.

5 Discussion

This paper represents an attempt to advance the long-standing efforts to understand a software programmer's productivity. The proposed research model is based on an integration of various independent productivity factors organized into a system of eight complementary factors with binary value ranges determining a programmer's productivity. This formulation enables us to derive two key isomorphic propositions, centered on the cognitive style of a programmer. Due to its tractability, the proposed model may guide empirical tests and provide several potential contributions. First, the model provides a novel and more comprehensive understanding of the productivity of a software programmer, in particular, and an information worker, in general. Second, the model may also contribute to the recent and growing literature on complementarities and their fit, especially as there are very few conceptions on the individual level and none that adopts a systems approach. Third, the proposed model may also contribute to the debate on the productivity paradox, particularly on how the use of IT systems may generate productivity gains at the level of an individual worker. Fourth, as the proposed research model draws on factors developed in various disciplinary quarters, such as psychology, managerial economics, organization studies, operations management, and information systems, this approach may contribute to each of these theoretical bodies with respect to how each adopted factor may be complemented with a unique set of other factors. Eventually, the proposed model may guide practitioners by suggesting which factors should be addressed and what is the nature of their synchronization.

The proposed model has several limitations. First, it focuses on the productivity of an individual programmer, while software may also be developed by teams of programmers. However, several studies argue that a comprehension of the team level requires an understanding of the individual level dynamics (Fong Boh et al., 2007; Sonnentag and Volmer, 2009). Second, we selected eight factors from a large set of established factors; we may have discarded some relevant and established factors. Third, the selection of factors was sourced in already established and documented studies, meaning that there always may be unknown elements, not previously surfaced by a study, that play a significant role, but are not accounted for in this paper. However, compared to the large set of studies that document the effects of single productivity factors, the proposed model offers a qualitative advancement by addressing multiple conditions affecting the productivity of programmers.

References

Abdolmohammadi, M. and A. Wright (1987). "An examination of the effects of experience and task complexity on audit judgments." *The Accounting Review* 62 (1), 1–13.

Agarwal, R. and J. Prasad (1999). "Are individual differences germane to the acceptance of new information technologies?." *Decision Sciences* 30 (2), 361–391.

Agor, W. H. (1984). *Intuitive Management*, Englewood Cliffs. NJ: Prentice Hall.

Ahearne, M., R. Jelinek and A. Rapp (2005). "Moving beyond the direct effect of SFA adoption on salesperson performance: Training and support as key moderating factors." *Industrial Marketing Management* 34 (4), 379–388.

Amabile, T. M. (1996). *Creativity in Context: Update to the Social Psychology of Creativity*. Boulder, CO: Westview Press.

Amabile, T. M., K. G. Hill, B. A. Hennessey and E. M. Tighe (1994). "The work preference inventory: Assessing intrinsic and extrinsic motivational orientations." *Journal of Personality and Social Psychology* 66 (5), 950–967.

Appelbaum, S. H., A. Marchionni and A. Fernandez (2008). "The Multi-tasking paradox: Perceptions, problems and strategies." *Management Decisions* 46 (9), 1313–1325.

Aral, S., E. Brynjolfsson and M. Van Alstyne (2012). "Information, technology and information worker productivity." *Information System Research* 23 (3), 849–867.

Athey, S. and S. Stern (2002). "The Impact of information technology on emergency health care outcomes." *RAND Journal of Economics* 33, 339–432.

Autor, D. H., F. Levy and R. J. Murnane (2003). "The skill content of recent technological change: An empirical exploration." *Quarterly Journal of Economics* 118 (4), 1279–1333.

Axtell, C. M., D. J. Holman, K. L. Unsworth, T. D. Wall, P. E. Waterson and E. Harrington (2000). "Shopfloor innovation: Facilitating the suggestion and implementation of ideas." *Journal of Occupational and Organizational Psychology* 73 (3), 265–285.

Baer, M., G. R. Oldham and A. Cummings (2003). "Rewarding creativity: When does it really matter?." *Leadership Quarterly* 14, 569–586.

Banker, R. D., S. M. Datarand and C. F. Kemerer (1987). "Factors affecting software maintenance productivity: An exploratory study." In: *Proceedings of the 8th International Conference on Information Systems*, Pittsburgh, Pennsylvania, 6-9 December, pp. 160–175.

Banker, R. D., S. M. Datar and C. F. Kemerer (1991). "A model to evaluate variables impacting the productivity of software maintenance projects." *Management Science* 37 (1), 1–18.

Baschab, J. and J. Piot (2007). *The Executive's Guide to Information Technology*. John Wiley and Sons.

Bell, C. S., D. R. Compeau and F. Olivera (2005). "Understanding the social implications of technological multitasking: A conceptual model." In: *Proceedings of the 4th Annual Workshop on HCI Research in MIS*. Las Vegas, NV, USA, pp. 80–84.

Benbasat, I. and A. S. Dexter (1985). "An experimental evaluation of graphical and color-enhanced information presentation." *Management Science* 31 (11), 1348–1364.

Berings, M. G. M. C., R. F. Poell and P. R. Simons (2005). "Conceptualizing on-the-job learning styles." *Human Resource Development Review* 4, 373–400.

Bourque, P. and R. E. Fairley (2014). *Guide to the Software Engineering Body of Knowledge, SWEBOK*. IEEE Computer Society Press.

Brown, L. L. (2001). "Review of the Kirton adaption-innovation inventory." *The Fourteenth Mental Measurements Yearbook*, 647–649.

Bruckhaus, T., N. H. Madhavji, I. Janssen and J. Henshaw (1996). "The Impact of tools on software productivity." *IEEE Software* 13 (5), 29–38.

Brynjolfsson, E. and P. Milgrom (2013). "Complementarity in organizations." In: *The Handbook of Organizational Economics*. Ed. by R. Gibbons and J. Roberts. Princeton University Press, pp. 11–55.

Burns, T. and G. M. Stalker (1961). *The Management of Innovation*. Tavistock, London.
Canfora, G., A. Cimitile, F. Garcia, M. Piattini and C. A. Visaggio (2007). "Evaluating performances of pair designing in industry." *Journal of Systems and Software* 80 (8), 1317–1327.
Capretz, L. F. (2003). "Personality types in software engineering." *International Journal of Human-Computer Studies* 58 (2), 207–214.
Carland, J. C., J. W. Carland, M. D. Ensley and H. W. Stewart (1994). "The implications of cognition and learning styles for management education." *Management Learning* 25 (3), 413–431.
Chen, I. J. and K. Popovich (2003). "Understanding customer relationship management (CRM) people, process and technology." *Business Process Management Journal* 9 (5), 672–688.
Clements-Croome, D. and Y. Kaluarachchi (2000). "Assessment and measurement of productivity." *Creating the Productive Workplace*, 129–166.
Clincy, V. A. (2003). "Software development productivity and cycle time reduction." *Journal of Computing Sciences in Colleges* 19 (2), 278–287.
Cockburn, A. and J. Highsmith (2001). "Agile software development: The people factor." *Computer* 11, 131–133.
Collins, H. (2003). *Enterprise Knowledge Portals: Next-Generation Portal Solutions for Dynamic Information Access, Better Decision Making, and Maximum Results*. AMACOM Div American Mgmt Assn.
Cougar, J. D. and R. A. Zawacki (1980). *Motivating and Managing Computer Personnel*. New York, NY: Wiley.
Czerwinski, M., E. Horvitz and S. Wilhite (2004). "A diary study of task switching and interruptions." In: *Proceedings of the ACM Human Factor in Computing Systems Conferenc*e, Vienna, Austria, pp. 175–182.
Daft R. L. (2000). *Management*. 5th ed. The Dryden Press, USA.
Darcy, D. P. and M. Ma (2005). "Exploring individual characteristics and programming performance: implications for programmer selection." In: *Proceedings of the 38th Annual Hawaii International Conference*, University of Maryland, College Park, pp. 314a–314a.
Das, A. and R. Narasimhan (2001). "Process-technology fit and its implications for manufacturing performance." *Journal of Operations Management* 19 (5), 521–540.
Davenport, T. H. (2005). *Thinking for a Living: How to Get Better Performance and Results from Knowledge Workers*. Boston: Harvard Business School Press.
de Barros Sampaio, S. C., E. A Barros, G. S. de Aquino and S. R. de Lemos Meira (2010). "A review of productivity factors and strategies on software development." In: *Proceedings of the 5th International Conference on Software Engineering Advances*, pp. 196–204.
DiMaggio, P. J. and W. W. Powell (1983). "The iron cage revisited: Institutional isomorphism and collective rationality in organizational fields." *American Sociological Review* 48 (2), 147–160.
Donaldson, S. E. and S. G. Siegel (2001). *Successful Software Development*. Prentice Hall Professional.
Drucker, P. F. (1999). "Knowledge-worker productivity: The biggest challenge." *California Management Review* 41 (2), 79–94.
Duff, A. (2004). "The Role of cognitive learning styles in accounting education: Developing learning competencies." *Journal of Accounting Education* 22 (1), 29–52.
Duncan, A. S. (1988). "Software development productivity tools and metrics." In: *Proceedings of the 10th International Conference on Software Engineering*, IEEE Computer Society Press, pp. 41–48.
Edberg, D. T. and B. J. Bowman (1996). "User-developed applications: An empirical study of application quality and developer productivity." *Journal of Management Information Systems* 13 (1), 167–185.
Ennen, E. and A. Richter (2010). "The whole is more than the sum of its parts – or is it? A review of the empirical literature on complementarities in organizations." *Journal of Managemen*t 36 (1), 207–233.
Fong Boh, W., S. A., Slaughter and J. A. Espinosa (2007). "Learning from experience in software

development: A multilevel analysis." *Management Science* 53 (8), 1315–1331.
Frey, B. S. and M. Osterloh (2002). *Successful Management by Motivation. Balancing Intrinsic and Extrinsic Incentives*. Berlin, Heidelberg, New York: Springer.
Hayes, J. and C. W. Allinson (1997). "Learning styles and training and development in work settings: lessons from educational research." *Educational Psychology: An International Journal of Experimental Educational Psychology* 17 (1-2), 185–193.
Helms-Mills, J., K. Dye and A. J. Mills (2008). *Understanding Organizational Change*. Routledge.
Henderson, D. R. (2007). *The Concise Encyclopedia of Economics*. Indianapolis, IN: Liberty Fund.
Hitt, L. M. and E. Brynjolfsson (1997). "Information technology and internal firm organization: An exploratory study." *Journal of Information Systems* 14 (2), 81–101.
Hunt, R. G., F. J. Krzystofiak, J. R. Meindl and A. M. Yousry (1989). "Cognitive style and decision making." *Organizational Behavior and Human Decision Processes* 44 (3), 436–453.
Jablokow, K. W. and D. E. Booth (2006). "The impact and management of cognitive gap in high performance product development organizations." *Journal of Engineering and Technology Management* 23 (4), 313–336.
Jain, V. and S. Kanungo (2005). "Beyond perceptions and usage: Impact of nature of information systems use on information system-enabled productivity." *International Journal of Human-Computer Interaction* 19 (1), 113–136.
Jennings, N. R. (2001). "An agent-based approach for building complex software systems." *Communications of the ACM* 44 (4), 35–41.
Jiang, Z. and P. Naudé (2007). "An examination of the factors influencing software development effort." *International Journal of Computer Information and Systems Science and Engineering* 1 (3), 182–191.
Jiang, Z., P. Naudé and C. Comstock (2007). "An investigation on the variation of software development productivity." *International Journal of Computer, Information, and Systems Sciences, and Engineering* 1 (2), 72–81.
Jones, C. (2000). *Software Assessments, Benchmarks, and Best Practices*. Addison-Wesley Longman Publishing Co., Inc.
Kasof, J., C. Chen, A. Himsel and E. Greenberger (2007). "Values and creativity." *Creativity Research Journal* 19, 105–122.
Keen, P. G. and M. S. Scott-Morton (1978). *Decision Support Systems: An Organizational Perspective*. Reading, MA: Addison-Wesley.
Kelly, A. (2008). *Changing Software Development: Learning to Become Agile*. John Wiley and Sons.
Kessels, J. W. (2001). "Learning in organizations: A corporate curriculum for the knowledge economy." *Futures* 33 (6), 497–506.
Kirikova, M., J. Grundspenkis, W. Wojtkowski, W.G. Wojtkowski, S. Wrycza and J. Zupancic (2012). *Information Systems Development: Advances in Methodologies, Components, and Management*. Springer Science and Business Media.
Kirton, M. J. (1976). "Adaptors and innovators: A description and measure." *Journal of Applied Psychology* 61 (5), 622–629.
Kirton, M. J. (1994). *Adaptors and Innovator.* 2nd Edition. Routledge, London.
Kock, N. F. Jr. and R. J. McQueen (1996). "Product flow, breadth and complexity of business processes: An empirical study of 15 business processes in three organizations." *Business Process Reengineering and Management Journal* 2 (2), 8–22.
Kozlowski, S. W. J. and B.S. Bell (2003). "Work groups and teams in organizations." In: *Comprehensive Handbook of Psychology.* Ed. by W. C. Borman, D. R. Ilgen, and R. J. Klimoski. New York: Wiley, pp. 333–375.
Kraiger, K. (2003). "Perspectives in training and development." In: *Handbook of Psychology.* Ed. by W.C. Borman, D.R. Ilgen, and R. Klimoski. New York: Wiley, pp. 171–192.
Lal, K. (2005). "In quest of the information sector: Measuring information workers for India." *Malaysian Journal of Library and Information Science* 10 (2), 85–104.

Laudon, K. C. and J. P. Laudon (2011). *Essentials of Management Information Systems*. (Upper Saddle River: Pearson.

Lokan, C. J. (2001). "Impact of subjective factors on software productivity." In: *Proceedings of 7th Australian Conference on Software Metrics, Melbourne*, pp. 29–30.

MacCormack, A., R. Verganti and M. Iansiti (2001). "Developing products on "Internet time": The autonomy of a flexible development process." *Management Science* 47 (1), 133–150.

Martin, R. C. (2003). *Agile Software Development: Principles, Patterns, and Practices*. Prentice Hall PTR.

McConnell, S. and C. Complete (1993). *A Practical Handbook of Software Construction*. Microsoft Press Redmond, WA, USA.

Milgrom, P. and J. Roberts (1995). "Complementarities and fit: Strategy, structure, and organizational change in manufacturing." *Journal of Accounting and Economics* 19, 179–208.

Mintzberg, H. (1979). *The Structuring of Organizations*. Prentice-Hall Inc., New Jersey, USA.

Mitchell, V. L. and R. W. Zmud (1999). "The effects of coupling IT and work process strategies in redesign projects." *Organization Science* 10 (4), 424–438.

North, K. and S. Gueldenberg (2011). *Effective Knowledge Work: Answers to the Management Challenges of the 21st Century*. Emerald Group Publishing.

Oldham, G. R. and A. Cummings (1996). "Employee creativity: Personal and contextual factors at work." *Academy of Management Journal* 39 (3), 607–634.

Ondrej, M., H. Jiri and H. Jan (2012). "Estimating productivity of software development using the total factor productivity approach." *International Journal of Engineering Business Management*, 4.

Parker, S. K., T.D. Wall and J. Cordery (2001). "Future work design research and practice: Towards an elaborated model of work design." *Journal of Occupational and Organizational Psychology* 74, 413–440.

Perry, G. M. (2002). *Absolute Beginner's Guide to Programming*. Que Publishing.

Petersen, K. (2011). "Measuring and predicting software productivity: A systematic map and review." *Information and Software Technology* 53, 317–343.

Pyöria, P. (2005). "The concept of knowledge work revisited." *Journal of Knowledge Management* 9 (3), 116–127.

Riding, R. and G. Douglas (1993). "The effect of cognitive style and mode of presentation on learning performance." *British Journal of Educational Psychology* 63 (2), 297–307.

Roberts, J. (2007). *The Modern Firm: Organizational Design for Performance and Growth*. Oxford University Press.

Rowe, A. J. and J. D. Boulgarides (1992). *Managerial Decision-making: A Guide to Successful Business Decision*. New York: Macmillan.

Rowe, A. J. and R. O. Mason (1987). *Managing with Style*. San Francisco: Jossey-Bass.

Sadler-Smith, E. and P. J. Smith (2004). "Strategies for accommodating individuals' styles and preferences in flexible learning programs." *British Journal of Educational Technology* 35, 395–412.

Scacchi, W. and D. Hurley (1995). "Understanding software productivity." *Software Engineering and Knowledge Engineering: Trends for the Next Decade* 4, 273–316.

Schneider, B. and D. B. Smith (2004). "Personality and organizational culture." In: *Personality and Organization,* B. Scheneider and D. B. Smith (eds.), Mahwah, New Jersey: Lawrence Erlbaum Associates, pp. 347–369.

Sharp, H., N. Baddoo, S. Beecham, T. Hall and H. Robinson (2009). "Models of motivation in software engineering." *Information and Software Technology* 51 (1), 219–233.

Silva, N. and R. Lopes (2012). "Independent assessment of safety-critical systems: We bring data!." In: *Software Reliability Engineering Workshops (ISSREW), 2012 IEEE 23rd International Symposium on*, pp. 84–84.

Sokoya, K. S. (2000). "Personal predictors of job satisfaction for the public sector managers: Implications for management practice and development in a developing company." *Journal of Business in Developing Nations* 4 (1).

Sonnentag, S. and J. Volmer (2009). "Individual-level predictors of task-related teamwork processes the role of expertise and self-efficacy in team meetings." *Group and Organization Management* 34 (1), 37–66.

Tausky, C. (1978). *Work Organizations: Major Theoretical Perspectives*. F. E. Peacock, Itasca Illinois.

Taylor, F. W. (1947). *Scientific Management.* New York: Harper and Row.

Tessem, B. (2011). "An Empirical study of decision making, participation, and empowerment in Norwegian software development organizations." In: *Agile Processes in Software Engineering and Extreme Programming*. Springer Berlin Heidelberg, pp. 253–265.

Toffolon, C. and S. Dakhli (2007). "A decision-oriented model of software engineering processes." In: *Proceedings European and Mediterranean Conference on Information Systems*, Polytechnic University of Valencia, pp. 24–26.

Tomaszewski, P. and L. Lundberg (2005). "Software development productivity on a new platform: An industrial case study." *Information and Software Technology* 47 (4), 257–269.

Wagner, S. and M. Ruhe (2008). "A systematic review of productivity factors in software development." In: *Proceedings of 2nd International Workshop on Software Productivity Analysis and Cost Estimation (SPACE 2008)*. State Key Laboratory of Computer Science, Institute of Software.

Weber, B. and W. Wild (2005). "Towards the agile management of business processes." *Professional Knowledge Management*. Springer Berlin Heidelberg, pp. 409–419.

Weber, B., M. Reichert and S. Rinderle-Ma (2008). "Change patterns and change support features–enhancing flexibility in process-aware information systems." *Data and Knowledge Engineering* 66 (3), 438–466.

Webster, M. (2012). "Bridging the information worker productivity gap: New challenges and opportunities for IT." *White Paper, IDC*.

Wolff, E. N. (2005). "The growth of information workers in the U.S. economy." *Communication of the ACM* 28 (10), 37–42.

Yazici, H. and R. Kluczny (1993). "Information display modes and user cognitive profiles: Interaction effects on the decision-making process." *Journal of Computer Information Systems*, 33 (4), 41–54.

Zabojnik, J. (2002). "Centralized and decentralized decision making in organizations." *Journal of Labor Economics* 20 (1), 1–22.

Zmud, R. W. (1979). "Individual differences and MIS success: A review of the empirical literature." *Management Science* 25 (10), 966–979.

Zuboff, S. (1988). *In the Age of the Smart Machine: The Future of Work and Power*. New York: Basic Books.