Information and Software Technology 56 (2014) 1537–1542

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Editorial

# Human factors in software development: On its underlying theories and the value of learning from related disciplines. A guest editorial introduction to the special issue

## 1. Introduction

Human factors play a very important role in Software Development [1]. According to Avison et al. [2] "Failure to include human factors may explain some of the dissatisfaction with conventional information systems development methodologies; they do not address real organizations" (p95 [2]). Software development has been characterized in essence as a human activity [3] where human factors play a critical role [4]. While the area of Human Factors spans a lot of different and diverse concepts and theories, the human factors aspects most often studied in software engineering research include coordination [5,6], collaboration in the development process [7–9], trust [10], expert recommendation [11], program comprehension [12], knowledge management [13,14] and culture [15].

The growing importance of human factors in software development research is clearly evidenced by the fact that the ICSE 2014 conference a track entirely devoted to Human factors, namely, "Social Aspects of Software Engineering". Furthermore, the 2014 ICSE conference keynote by James Herbsleb [16] presented the theory of socio-technical coordination and represented a call for further development of theories on coordination in Software Engineering (SE). In this editorial we not only reiterate this call, but also suggest SE researchers to draw on reference disciplines such as the field of Information Systems to borrow well-established theories.

In the next section we first present empirical evidence on the importance of research involving Human Factors in the field of Software Engineering. We then run a citation analysis exercise to identify the prominent theories related to Human Factors in SD.

## 2. Importance of human factors

In examining the importance of Human Factors in SD as a research topic, we conducted a citation analysis. We first looked for systematic reviews previously published on the topic. We found that though Software Engineering (SE) and Information Systems (IS) communities have carried out systematic reviews that are related to the special issue [17–19], they have not covered all possible human factors. Each of these reviews has investigated HF from a particular angle, e.g. Beecham et al. [17] looked at HFs from the perspective of motivation of SE professionals. The only comprehensive literature review that directly focused on HFs in SD is the one of Pirzadeh [20].

We looked into the search terms used in these reviews when searching for relevant primary studies to include in their reviews. Pirzadeh [20] uses the following base search term to locate literature on Human factors in SD:

("Software Development Process" OR "Software Development" OR "Software Engineering") AND ("Human factors" OR "Human issues")

However, when using this search expression in Web of Science (ISI), we noticed it returned far fewer articles than we thought were available. So we broadened the search string using the popular topics on Human Factors [17] to arrive at the following expression:

("Software Engineering" OR "Software Development") AND (Coordination OR Collaboration OR Communication OR "Human Factors" OR "Human Issues" OR "Task Allocation" OR "Task Distribution" OR Trust OR Knowledge OR Motivation OR Expert OR Culture "User Participation" OR "User Involvement" OR psychology OR "psychological factors" OR motivator* OR prefer* OR behavio*r*)

Though this search expression would definitely not return all the articles related to Human Factors in SD, we can be sure that we include most of the popular Human Factor topics. For locating the number of software engineering papers published over the last decade (and until 2014), we used the following expression:

("Software Engineering" OR "Software Development")

This expression is sufficient as the search term "Software Development" makes the search term "Software Development Process" [20] redundant.

We then searched for various human factors related to software engineering in the Web of Science (ISI) research database using the above search expression. Though this expression does return a few false positives (papers that only marginally refer to human factors), one can assume that the number of those false positives is constant for each year over the last decade.

In Fig. 1 we plotted the total number of software engineering articles as well as the total number of articles related to Human Factors in software development.

http://dx.doi.org/10.1016/j.infsof.2014.07.006

We then calculated the percentage of papers related to Human Factors in SD among the total number of software engineering papers published each year. Fig. 2 shows an upward trend in the percentage of papers that deal with Human Factors in SD. This upward trend is an indication of an increasing number of papers on HF in SD being written and hence being published. We think this upward trend is not surprising as more researchers doing empirical research in Software Engineering are seeing the importance of addressing the issues of Human Factors [21]. We next identify the main theories in Human Factors in SD over the last decade.

## 3. Identifying the main theories in human factors in software development

Theory is generally considered to be the bedrock of a discipline. In recent years, theory, as understood in the field of sciences, has gained popularity in the field of empirical SE [21–23]. Members of the International Empirical Software Engineering Research Network (ISERN) engaged increasingly more and more into research initiatives that aim to formulate theories, evaluating theories and evaluating the use of theories in the field [29]. For example, Endres and Rombach [24] catalogued theories in the various sub-fields of SE, while searching "for the fundamental rules in SE, why they work and why they are useful". Also, empirical SE researchers have made calls for borrowing theories from other disciplines [25] and applying them to explain SE phenomena [26] and for improving our reasoning about generalization of empirical claims in SE studies [26]. The most recent indicator for this increased interest in SE theories is the SEMAT initiative (www.semat.org) [27] that aims to increase the focus on general theories in SE. We can conclude that there is an agreement in the empirical SE community that understanding the primary theories of a field (such as SE) or a subfield (such as Human Factors in SD) provides insight into the distinct evolutionary phases of its development [28]. In order to look into the theories used in empirical SE studies on HF, we need to state what we mean by theory for the purpose of this editorial and our special issue.

Two primary issues emerge with regard to identifying theories related to a field; namely, (i) how do we define a theory and what is its structure? And, (ii) how does one identify those theories?

Regarding the first question, there is little consensus among researchers about what constitutes a theory. While Sutton and Staw [29] and Weber [30] take the harder line, ascribing the term theory to models that both explain and predict, Weick [31] describes the premature theoretical frameworks and models as

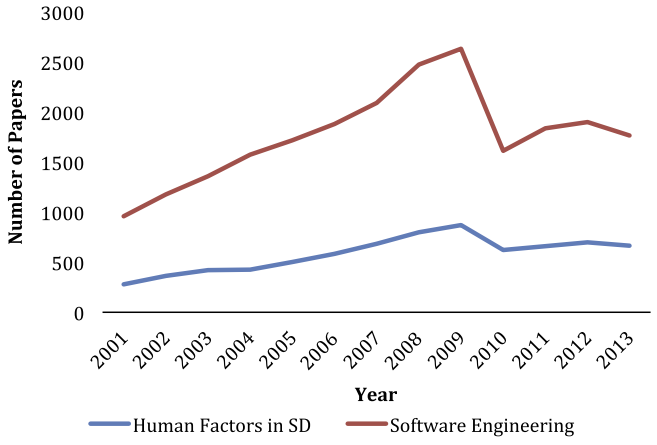


**Fig. 1.** The total number of Software Engineering and Human Factors in SD articles published between 2000 and 2014.

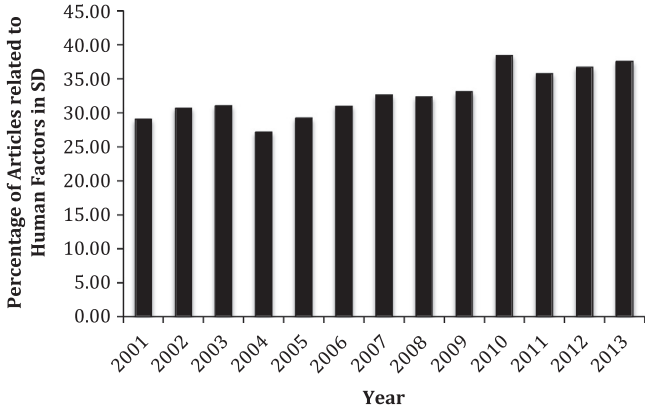


**Fig. 2.** The change in the percentage of papers published on Human Factors in SD over the last decade.

theorizing and deems them to be an essential part of theory formation. For example, Gregor [32] defines theory rather broadly as:

> ("... *abstract entities that aim to describe, explain, and enhance understanding of the world and, in some cases, to provide predictions of what will happen in the future and to give a basis for intervention and action*" (p616 [32])).

She further describes a classification of theories into types I–V;

- type I – theories for analyses,
- type II – theories for explanation,
- type III – theories for prediction,
- type IV – theories for explanation and prediction,
- type V – theories for design and action.

Sjøberg et al. [33] define the elements of theory to consist of (i) Entities – the main constructs of the theory (ii) Relationships among the entities (iii) the Reason behind the relationships (this provides the explanatory power), and (iv) Scope conditions of the relationships. Wieringa and Daneva [26] consider that theories need to have a (i) conceptual framework, (ii) generalizations and/or models, (iii) scope conditions need to be defined, and optionally (iv) causal explanations.

The structure of a theory is dependent on the field the theory belongs to [32,33]. Human Factors in SD can be considered a subfield of Empirical Software Engineering, and hence the theory structure adopted by Sjøberg et al. [33] (and related to Wieringa and Daneva [26]) is applicable. We hence adopt Sjøberg et al.'s [33] definition of theory, in order to briefly analyse the essential theories related to Human Factors in SD later on in this section.

To answer the second question on theory identification, we employed a qualitative citation analysis technique as done by Moody et al. [34]. This technique effectively complements traditional quantitative citation analysis approaches such as identifying reference disciplines [35], identifying citation classics [36] and identifying intellectual communities [37].

We analyzed the articles indexed in Web of Science database published in the top 6 leading software engineering journals: TSE, TOSEM, JSS, SPE, IEEE Software and IST according to [35,38]. We collected all the articles from these journals published over the last complete decade (2001–2010). We left out EMSE, as Web of Science began indexing EMSE only since 2003. The reason we chose to analyse the references of all the articles, and not just articles related to human factors since articles that do not explicitly deal with human factors could still mention them in passing and then use human factor references, that they consider important, to substantiate their claims. We think this method is superior to

ranking articles based on the total number of citations in Web of Science, as Web of Science (ISI) only provides the citations by articles that are indexed by the database [34].

We extracted the reference lists from all the papers and collated them to get the cumulative list of all papers referred to by the most important papers in the discipline. We ended with about 90,000 scientific works, with many of the books/articles being cited only once. We arranged the articles in descending order of the number of citations and among these 90,000 books/articles we concentrated on the top 500 to locate the primary theories. Table 1 represents the top 10 most cited articles/books in our corpus (in the HistCite output format). The 2nd and 6th most cited works (in grey) deal with Human Factors in SD.

In a similar fashion we combed through the top 500 to obtain the list of articles and books in Table 2.

The process of gathering citations was not very straightforward, as some of the references had slightly different spellings and style. So we used a citation analysis tool called HistCite [53] to collate the references from Web of Science. Despite the usage of the tool there were some references not added to the total. We did this manually, by sorting the references by name and then adding the references to the total. We included only those works that explicitly mentioned human factor constructs and relations and hence left out seminal works such as the modularization work of Parnas [54] (which came in the 3rd place with 48 citations), as it only has an influence on human factors, through task allocation, and does not explicitly deal with human factors. We also left out the book by Coplien and Schmidt [55] (that ranked 10th with 19 citations), as a more in-depth analysis of its citations revealed that the articles cited programming patterns rather than the chapter on Coplien's organizational patterns [56]. To ensure the validity of the identification, the first two authors of this editorial did the analysis separately and then we compared and consolidated the findings. In Table 2, we see the 20 most cited articles/books that are related to Human Factors in SD in the last full decade (2001–2010). From Table 2, we see a healthy mix of books (8 of the 20) [34]. We also see a dominance of works on Project Management (including Open source and Global Project Management), and in fact, one can even argue that all the books/articles in Table 2 are related to Project Management in software development. Though some of the works deal with specific issues in Software Project Management like Project Estimation [39,41,43] and to specific development environments like Global software development [15,51] or Open source development environment [8]. On analysing the works, by looking for Human Factor theories, we arrive at Table 3 below. In Table 3 we identify the native (from SE) and imported (from reference disciplines like social sciences) Human Factor theories as well as the type of the theory. We see that among the top 20 most cited works only 10 are theories (the shaded rows), with well-defined entities, relationships as well as the reasoning and scope of the relationships [33].

We decided if a theory (in an article) is native to SE or imported from a reference discipline based on whether the source journal belonged to the SE field [35]. We now briefly describe the theories identified in Table 3.

**Table 1**
The top 10 most cited works in our corpus.

| Article/book | # Citations |
|---|---|
| Gamma E., 1995, DESIGN PATTERNS ELEM | 172 |
| Boehm Barry W., 1981, SOFTWARE ENG EC | 135 |
| Chidamber S.R., 1994, IEEE T SOFTWARE ENG | 128 |
| Fowler M., 1999, REFACTORING IMPROVIN | 74 |
| Shaw M., 1996, SOFTWARE ARCHITECTUR | 73 |
| Brooks F., 1995, MYTHICAL MAN MONTH E | 73 |
| Booch G., 1999, UNIFIED MODELING LAN | 66 |
| Beck K., 2000, EXTREME PROGRAMMING | 65 |
| Gamma E., 1994, DESIGN PATTERNS ELEM | 63 |
| Bosch J., 2000, DESIGN USE SOFTWARE | 63 |

The books of Boehm [39] and Boehm et al., 2000 [41] are well-known predictive theories on estimating the costs of software projects. These theories use human factors in the predictive models, particularly, the personal characteristics of the software development staff, such as analyst capability, applications experience, programming language experience, to name a few [41].

One can argue that many of the articles related to various 'Best Practices' in Table 3 are actually theories in the framework of Sjøberg et al. [33]. However, the definition and scope of the relationships among their constructs is clearer in some cases (e.g. [57,58]) than in others. Here we note that we not only consider the works themselves, but also the derivative works published later. The 'Best Practices' themselves contain the basis of theories, or as Weick [31] puts it, the authors of these works have done a very good job at theorizing [31]. We therefore would like to distinguish these articles from well-established theories such as Brooks' Law [40] and Conway's Law [50]. Brooks' Law ("*Adding manpower to a late software project makes it later*") (p137 [40]), has been tested in various situations and its scope has been refined in literature [57,59]. The same holds true for Conway's Law ("*...organizations which design systems... are constrained to produce designs which are copies of the communication structures of these organizations*") (p31 [50]), or the 'Mirroring Hypothesis' as it is better known in the field of Industrial Engineering [58,60,61]. On the other hand, we also see imported theories from reference disciplines in the list, e.g. the well-established Technology Acceptance Model (TAM) of Davis [44] from the IS discipline, the classic knowledge-based theory of the firm by Nonaka and Takeuchi [49] from the IS/Knowledge management discipline, and the risk management theory of Lyytinen et al. [52]. The other native theories that stand out in the list are the explanatory theory of the Open Source development process by Mockus et al. [8], theory on factors that influence successful coordination by Kraut and Streeter [5] and the theory on program comprehension by Brooks [12].

## 4. Discussion and conclusion

Human Factors in SD as described earlier could be considered as a sub-field of Empirical SE and hence shares methodological issues with social and behavioural sciences [33]. One would hence expect many of the main theories in Human Factors in SD to come from a field related to behavioural and social sciences like Information Systems. However, what we observe in Table 3 is that seven of the ten theories that we identified are native to the SE discipline. At first glance, this observation may seem surprising, however we think it is traceable to a 2004 finding of Glass et al. [35] who describe how the fields of Computer Science (CS) and SE are *inward looking* and do not borrow theories and concepts from other reference disciplines. According to Weber [62], this is good for the discipline (or sub-discipline), as he suggests that for a disciple to survive in the long run, it should pursue research on paradigms (or theories [34]) belonging to the discipline [62]. Despite this and the historical roots and traditions in CS and SE, in the specific case of Human Factors in SD, we think that it is justifiable and worthwhile for empirical SE researchers to borrow from the wealth of knowledge in the social and behavioural sciences. First, such theories are good for informing the design of new empirical studies, which might have the long term benefit of helping the research literature converge on a particular topic (or research question), as Stol and Fitzgerald [23] indicate. Second, borrowed theories can contribute to the development of SE-specific frameworks potentially useful to empirical SE researchers for organizing existing concepts from the literature, or in assisting in the development

**Table 2**
Top 20 works related to Human Factors in SD in the top 500 most cited works from the previous decade.

| Rank | Reference | Subfield | Book (B)/article (A) | # Citations |
|---|---|---|---|---|
| 1 | Boehm B., (1981) [39] | Cost Estimation | B | 135 |
| 2 | Brooks F., (1995) [40] | Project Management | B | 73 |
| 3 | Boehm B. et al., (2000) [41] | Cost Estimation | B | 39 |
| 4 | Cockburn A., (2002) [42] | Agile Software Development | B | 38 |
| 5 | Jorgensen M., (2004) [43] | Expert Estimation | A | 38 |
| 6 | Curtis B. et al., (1988) [1] | Project Management | A | 35 |
| 7 | Davis F.D., (1989) [44] | Technology Acceptance | A | 33 |
| 8 | Brooks F.P., (1987) [45] | Project Management | A | 24 |
| 9 | Mockus A. et al., (2002) [8] | Open Source Software Development | A | 22 |
| 10 | DeMarco T. and Lister T., (1999) [46] | Project Management | B | 22 |
| 11 | Carmel E., (1999) [47] | Global Project Management | B | 17 |
| 12 | Humphrey W.S., (2000) [48] | Software development process | B | 17 |
| 13 | Nonaka I. and Takeuchi H., (1995) [49] | Organizational Knowledge Management | B | 15 |
| 14 | Kraut R.E. and Streeter L., (1995) [5] | Coordination in software development | A | 15 |
| 15 | Brooks R., (1983) [12] | Program Comprehension | A | 13 |
| 16 | Carmel E. and Agarwal R., (2001) [15] | Global Project Management | A | 12 |
| 17 | Conway M.E., (1968) [50] | Coordination in software development | A | 11 |
| 18 | Herbsleb J.D. and Moitra, (2001) [51] | Global Project Management | A | 10 |
| 19 | Lyytinen K. et al., (1998) [52] | Software development risk management | A | 10 |
| 20 | Alavi and Leidner [19] | Knowledge Management | A | 9 |

**Table 3**
The list of most cited works related to Human Factors in SD and the type of theorizing/theories they represent. Where type I are theories for analyses, type II – theories for explanation, type III are theories for prediction, type IV are theories for explanation and prediction, type V are theories for design and action.

| Rank | Reference | Theory type | Theory description |
|---|---|---|---|
| 1 | Boehm B., 1981 [39] | III | Native theory |
| 2 | Brooks F., 1995 [40] | IV | Native theory |
| 3 | Boehm B. et al., 2000 [41] | III | Native theory |
| 4 | Cockburn A., 2002 [42] | – | Best Practices |
| 5 | Jorgensen M., 2004 [43] | – | Structured Lit. Review |
| 6 | Curtis B. et al., 1988 [1] | – | Experience Report |
| 7 | Davis F.D., 1989 [44] | IV | Imported theory (IS) |
| 8 | Brooks F.P., 1987 [45] | – | Experience Report |
| 9 | Mockus A. et al., 2002 [8] | II | Native theory |
| 10 | DeMarco T. and Lister T., 1999 [46] | – | Best Practices |
| 11 | Carmel E., 1999 [47] | – | Best Practices |
| 12 | Humphrey W.S., 2000 [48] | – | Best Practices |
| 13 | Nonaka I. and Takeuchi H., 1995 [49] | III | Imported Theory (KM) |
| 14 | Kraut R.E. and Streeter L., 1995 [5] | IV | Native theory |
| 15 | Brooks R., 1983 [12] | IV | Native theory |
| 16 | Carmel E. and Agarwal R., 2001 [15] | – | Best Practices |
| 17 | Conway M.E., 1968 [50] | IV | Native Theory |
| 18 | Herbsleb J.D., 2001 [51] | – | Best Practices |
| 19 | Lyytinen K. et al., 1998 [52] | IV | Imported theory (IS) |
| 20 | Alavi and Leidner [19] | – | Structured Lit. Review |

and evaluation of knowledge claims, e.g. hypotheses. Third, borrowing theories from other disciplines can increase the explanatory power of the theory – explaining the constructs and propositions of a theory [21,33]. For example, many of the core theories in IS field are related to Human Factors in SD ([34] see p9). Interesting and relevant theories in the IS discipline can include Unified Theory of Acceptance and Use of Technology UTAUT [63], Task-technology fit theory [64], Transactive memory theory [65], Adaptive structuration theory [66], among others. A longer list of theories from the IS domain can be found at the IS theory page [67]. The papers by Licorish and MacDonell, and the one of Barzilay and Urquhart, included in this Special Issue, provide examples of theory usage from other fields – social psychology, psycholinguistics, to study HF related phenomena in SE.

## 5. The papers in this special issue

We received a total of 23 research papers. All went through a rigorous round of review that brought us to 5 accepted papers. Below, we briefly describe the research in these papers.

Daniel Schall in his paper "*Who to follow recommendation in large-scale online development communities*" describes an approach that recommends relevant users in a social media platform and evaluates this approach on GitHub while implicitly using a design science approach. The author also considers the effect of information propagation in follower networks to random or strategic node removal. The key finding is that a link-based algorithm can be used for context sensitive follow recommendations. They also find that, as GitHub is sensitive to authority based node removal, the information flow through follower relations would be impacted by specific authority node removal.

Maria Paasivaara and Casper Lassenius in their paper "*Communities of practice in a large distributed agile software development organization—Case Ericsson*" explicate the human factors involved when a large software development organization transitions from a plan-driven to an agile approach. Their case study describes how one particular company, namely Ericsson, adopted Communities of Practice – a group of experts who share a common interest on a particular topic, can help in knowledge sharing, inter-team coordination and also in mitigating problems during the Agile

transformation. The authors find that success factors of well-functioning CoPs include among others having a passionate leader, a proper agenda, an open community and cross-site participation when needed.

Sherlock Licorish and Stephen MacDonell in their paper "*Understanding the attitudes, knowledge sharing behaviours and task performance of core developers: A longitudinal study*", investigate the way in which core developers' attitudes, knowledge sharing behaviours and task performance change over the course of a software development project. The authors analyse and mine repositories of ten software project teams of the Jazz project. They use social psychology and psycholinguistic theories to find that the attitudes and involvement in knowledge sharing of core developers is linked to the demands of the wider teams.

Mika Mäntylä and Juha Itkonen in their paper "*How are software defects found? The role of implicit defect detection, individual responsibility, documents, and knowledge*" survey people in four development organizations to understand how the perspectives of responsibility, activity, knowledge and document use affect defect detection. They find that developers find more defects through implicit defect detection (activities performed while doing tasks other than testing and code reviews) than through explicit defect detection (formal testing and reviews).

Ohad Barzilay and Cathy Urquhart in the paper "*Understanding reuse of software examples: A case study of prejudice in a community of practice*" examine the human factors that determine the usage of examples among software developers through an online survey. To understand the code example usage, the authors use the *prejudice theory* as a theoretical lens. They highlight concerns and identify additional barriers of software developers regarding reuse of code examples.

## Acknowledgements

The authors would like to thank the program committee of the ECIS 2013 track on "Methods, Tools and Human Factors in IS/IT Development and Management", many of whom also served as reviewers for this special issue. We would especially like to thank the following 58 reviewers (in alphabetical order): Paris Avgeriou, Mehmet N. Aydin, Ohad Barzilay, Sarah Beecham, Richard Berntsson-Svensson, Daniel M. Berry, Luigi Buglione, Andrea Capiluppi, Nelly Condori-Fernandez, Daniela Cruzes, Renata Mendes de Araujo, Matthijs den Besten, Siva Dorairaj, Leticia Duboc, Christof Ebert, Yeliz Eseryel, Cigdem Gencel, Marcela Genero, Andrea Herrmann, Rashina Hoda, Martin Höst, Netta Iivari, Sami Jantunen, Philippe Kruchten, Irwin Kwan, Casper Lassenius, Luigi Lavazza, David Lizcano, Kalle Lyytinen, Stephen MacDonell, Christina Manteli, Mika Mäntylä, Sabrina Marczak, Sanjay Misra, Nils Brede Moe, Ana M. Moreno, Barbara Paech, Ricardo Colomo Palacios, Maria Paasivaara, Kai Petersen, Dietmar Pfahl, Norah Power, Rudolf Ramler, Björn Regnell, Barbara Russo, Daniel Schall, Norbert Seyff, Helen Sharp, Klaas Sikkel, Darja Smite, Miroslaw Staron, Damian Tamburri, Gitte Tjørnehøj, Marco Torchiano, Patrick Wagstrom, Norihiro Yoshida and Zhemin Zhu.

We extend our special thanks for the IST editor in chief, Claes Wohlin for his advice and prompt response in dealing with all the practical aspects of the special issue.

## References

[1] B. Curtis, H. Krasner, N. Iscoe, A field-study of the software-design process for large systems, Commun. ACM 31 (November) (1988) 1268–1287.
[2] D.E. Avison, F. Lau, M.D. Myers, P.A. Nielsen, Action research, Commun. ACM 42 (1999) 94–97.
[3] C. Ebert, R. Dumke, Software Measurement, Springer, 2007.
[4] C. Amrit, Improving Coordination in Software Development through Social and Technical Network Analysis, PhD Thesis, University of Twente, Enschede, 2008.
[5] R.E. Kraut, L.A. Streeter, Coordination in software development, Commun. ACM 38 (1995) 69–81.
[6] C. Amrit, J. van Hillegersberg, Detecting coordination problems in collaborative software development environments, Inform. Syst. Manage. 25 (2008) 57–70.
[7] J.D. Herbsleb, A. Mockus, An empirical study of speed and communication in globally distributed software development, IEEE Trans. Softw. Eng. 29 (June) (2003) 481–494.
[8] A. Mockus, R.O.Y.T. Fielding, J.D. Herbsleb, Two case studies of open source software development: Apache and Mozilla, ACM Trans. Softw. Eng. Methodol. 11 (2002) 309–346.
[9] C. Amrit, J. van Hillegersberg, Exploring the impact of socio-technical core-periphery structures in open source software development, J. Inform. Technol. 25 (2010) 216–229.
[10] R. Sabherwal, The role of trust in outsourced IS development projects, Commun. ACM 42 (1999) 80–86.
[11] D. Cubranic, G.C. Murphy, J. Singer, K.S. Booth, Hipikat: a project memory for software development, IEEE Trans. Softw. Eng. 31 (2005) 446–465.
[12] R. Brooks, Towards a theory of the comprehension of computer programs, Int. J. Man Mach. Stud. 18 (1983) 543–554.
[13] M. Lindvall, I. Rus, Knowledge management in software engineering, IEEE Softw. 19 (2002) 0026–38.
[14] J.A. Espinosa, S.A. Slaughter, R.E. Kraut, J.D. Herbsleb, Team knowledge and coordination in geographically distributed software development, J. Manage. Inform. Syst. 24 (2007) 135–169.
[15] E. Carmel, R. Agarwal, Tactical approaches for alleviating distance in global software development, IEEE Softw. 18 (2001) 22–29.
[16] J. Herbsleb, Socio-technical coordination (keynote), Presented at the Companion Proceedings of the 36th International Conference on Software Engineering, Hyderabad, India, 2014.
[17] S. Beecham, N. Baddoo, T. Hall, H. Robinson, H. Sharp, Motivation in software engineering: a systematic literature review, Inf. Softw. Technol. 50 (2008) 860–878.
[18] F.O. Bjørnson, T. Dingsøyr, Knowledge management in software engineering: a systematic review of studied concepts, findings and research methods used, Inf. Softw. Technol. 50 (2008) 1055–1068.
[19] M. Alavi, D.E. Leidner, Review: knowledge management and knowledge management systems: conceptual foundations and research issues, MIS Quart. (2001) 107–136.
[20] L. Pirzadeh, Human Factors in Software Development: A Systematic Literature Review, Master Thesis, Chalmers University of Technology, 2010.
[21] J.E. Hannay, D.I. Sjoberg, T. Dyba, A systematic review of theory use in software engineering experiments, IEEE Trans. Softw. Eng. 33 (2007) 87–107.
[22] P. Johnson, M. Ekstedt, I. Jacobson, Where's the theory for software engineering?, IEEE Softw 29 (2012) 96.
[23] K.-J. Stol, B. Fitzgerald, Uncovering theories in software engineering, in: 2013 2nd SEMAT Workshop on a General Theory of Software Engineering (GTSE), 2013, pp. 5–14.
[24] A. Endres, H.D. Rombach, A Handbook of Software and Systems Engineering: Empirical Observations, Laws and Theories, Pearson Education, 2003.
[25] S.E. Sim, J. Singer, M.-A. Storey, Beg, borrow, or steal: using multidisciplinary approaches in empirical software engineering research, Empir. Softw. Eng. 6 (2001) 85–93.
[26] R. Wieringa, M. Daneva, Five strategies for generalizing software engineering theories, Sci. Comput. Program. (2014) (in press).
[27] P. Ralph, P. Johnson, H. Jordan, Report on the first SEMAT workshop on general theory of software engineering (GTSE 2012), ACM SIGSOFT Softw. Eng. Notes 38 (2013) 26–28.
[28] T.S. Kuhn, The Structure of Scientific Revolutions, University of Chicago Press, 2012.
[29] R.I. Sutton, B.M. Staw, What theory is not, Admin. Sci. Quart. (1995) 371–384.
[30] R. Weber, Evaluating and developing theories in the information systems discipline, J. Assoc. Inform. Syst. 13 (2012) 1–30.
[31] K.E. Weick, What theory is not, theorizing is, Admin. Sci. Quart. (1995) 385–390.
[32] S. Gregor, The nature of theory in information systems, MIS Quart. 30 (2006) 611–642.
[33] D.I. Sjøberg, T. Dybå, B.C. Anda, J.E. Hannay, Building theories in software engineering, in: Guide to Advanced Empirical Software Engineering, Springer, 2008, pp. 312–336.
[34] D. Moody, M.E. Iacob, C. Amrit, In search of paradigms: identifying the theoretical foundations of the IS field, Presented at the European Conference on Information Systems (ECIS), 2010.
[35] R.L. Glass, V. Ramesh, I. Vessey, An analysis of research in computing disciplines, Commun. ACM 47 (2004) 89–94.
[36] K.A. Walstrom, L.N. Leonard, Citation classics from the information systems literature, Inform. Manage. 38 (2000) 59–72.
[37] K.R. Larsen, D.E. Monarchi, D.S. Hovorka, C.N. Bailey, Analyzing unstructured text data: using latent categorization to identify intellectual communities in information systems, Decis. Support Syst. 45 (2008) 884–896.
[38] T. Tse, T.Y. Chen, R.L. Glass, An assessment of systems and software engineering scholars and institutions (2000–2004), J. Syst. Softw. 79 (2006) 816–819.
[39] B.W. Boehm, Software Engineering Economics (1981).
[40] F.P. Brooks, The Mythical Man Month, Prentice Hall, 1975.
[41] B.W. Boehm, R. Madachy, B. Steece, Software Cost Estimation with Cocomo II with Cdrom, Prentice Hall PTR, 2000.
[42] A. Cockburn, Agile Software Development: The Cooperative Game, Pearson Education (2006).

[43] M. Jørgensen, A review of studies on expert estimation of software development effort, J. Syst. Softw. 70 (2004) 37–60.
[44] F.D. Davis, Perceived usefulness, perceived ease of use, and user acceptance of information technology, MIS Quart. (1989) 319–340.
[45] J. Frederick, P. Brooks, No silver bullet essence and accidents of software engineering, Computer 20 (1987) 10–19.
[46] T. DeMarco, T. Lister, Peopleware: Productive Projects and Teams, second ed., Dorset House Publishing Co., Inc., 1999.
[47] E. Carmel, Global Software Teams: Collaborating across Borders and Time Zones, Prentice Hall PTR, 1999.
[48] W.S. Humphrey, Introduction to the Team Software Process (sm), Addison-Wesley Professional, 2000.
[49] I. Nonaka, H. Takeuchi, The Knowledge-creating Company: How Japanese Companies Create the Dynamics of Innovation, Oxford University Press, 1995.
[50] M. Conway, How do committees invent, Datamation 14 (1968) 28–31.
[51] J.D. Herbsleb, D. Moitra, Global software development, IEEE Softw. 18 (2001) 16–20.
[52] K. Lyytinen, L. Mathiassen, J. Ropponen, Attention shaping and software risk—a categorical analysis of four classical risk management approaches, Inform. Syst. Res. 9 (1998) 233–255.
[53] E. Garfield, S. Paris, W.G. Stock, HistCiteTM: A Software Tool for Informetric Analysis of Citation Linkage.
[54] D.L. Parnas, On the criteria to be used in decomposing systems into modules, Commun. ACM 15 (1972) 1053–1058.
[55] J.O. Coplien, D.C. Schmidt, Pattern Languages of Program Design, New York, NY, USA, 1995.
[56] J.O. Coplien, A development process generative pattern language, in: Proceedings of PLoP/94, ed. Monticello, IL, 1994, pp. 1–33.
[57] T.K. Abdel-Hamid, The dynamics of software project staffing: a system dynamics based simulation approach, IEEE Trans. Softw. Eng. 15 (1989) 109–119.
[58] M.E. Sosa, S.D. Eppinger, C.M. Rowles, The misalignment of product architecture and organizational structure in complex product development, J. Manage. Sci. 50 (2004) 1674–1689.
[59] L. Williams, A. Shukla, A.I. Anton, An initial exploration of the relationship between pair programming and Brooks' law, in: Agile Development Conference, 2004, pp. 11–20.
[60] L. Colfer, C.Y. Baldwin, The Mirroring Hypothesis: Theory, Evidence and Exceptions, Harvard Business School, 2010. pp. 10-058.
[61] M. Cataldo, P. Wagstrom, J.D. Herbsleb, K.M. Carley, Identification of coordination requirements: implications for the design of collaboration and awareness tools, Presented at the Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work, Banff, Alberta, Canada, 2006.
[62] R. Weber, Toward a theory of artifacts: a paradigmatic base for information systems research, J. Inform. Syst. 1 (1987) 3–19.
[63] V. Venkatesh, M.G. Morris, G.B. Davis, F.D. Davis, User acceptance of information technology: toward a unified view, MIS Quart. (2003) 425–478.
[64] D.L. Goodhue, R.L. Thompson, Task-technology fit and individual performance, MIS Quart. (1995) 213–236.
[65] D.M. Wegner, Transactive memory: a contemporary analysis of the group mind, in: Theories of Group Behavior, Springer, 1987, pp. 185–208.
[66] G. DeSanctis, M.S. Poole, Capturing the complexity in advanced technology use: adaptive structuration theory, Organ. Sci. 5 (1994) 121–147.
[67] K.R. Larsen, G. Allen, A. Vance, D.E. Eargle, Theories Used in IS Research Wiki, 2014, June. <http://istheory.byu.edu>.

Chintan Amrit
*IEBIS Department, University of Twente, The Netherlands*
*E-mail address:* c.amrit@utwente.nl

Maya Daneva
*Services, Cyber-security and Safety, University of Twente, The Netherlands*
*E-mail address:* m.daneva@utwente.nl

Daniela Damian
*Department of Computer Science, University of Victoria, Canada*
*E-mail address:* danielad@cs.uvic.ca

Available online 23 July 2014