Personalised Continuous Software Engineering

Efi Papatheocharous
SICS Swedish ICT AB
Stockholm, Sweden
+46 70 528 63 76
efi.papatheocharous@sics.se

Marios Belk
University of Cyprus
Department of Computer Science
Nicosia, Cyprus
belk@cs.ucy.ac.cy

Jaana Nyfjord
SICS Swedish ICT AB
Stockholm, Sweden
+46 72 232 68 03
jaana.nyfjord@sics.se

Panagiotis Germanakos
SAP AG
Walldorf, Germany
+49 (0) 62 277 426 96
panagiotis.germanakos@sap.com

George Samaras
University of Cyprus
Department of Computer Science
Nicosia, Cyprus
cssamara@cs.ucy.ac.cy

ABSTRACT
This work describes how human factors can influence continuous software engineering. The reasoning begins from the Agile Manifesto promoting individuals and interactions over processes and tools. The organisational need to continuously develop, release and learn from software development in rapid cycles requires empowered and self-organised agile teams. However, these teams are formed without necessarily considering the members' individual characteristics towards effective teamwork, from the personality and cognitive perspective. In this realm, this paper proposes a two level approach: first, form teams based on their collective personality traits and second, provide personalised tools and methods based on their individual differences in cognitive processing. The approach is motivated by a study conducted in a business environment focusing on task execution, satisfaction and effectiveness of team members in relation to their personalities and cognitive characteristics. Our preliminary results show that human factors provide a promising basis for increasing the capability of continuous software engineering.

Categories and Subject Descriptors
D.2.9 [**Software Engineering**]: Management

General Terms
Management, Measurement, Performance, Experimentation, Human Factors.

Keywords
Agile, human factors, personality, cognitive factors.

1. INTRODUCTION
Agile methods in software development have become common practice in organisations today. One of the main ideas in the Agile Manifesto for software development emphasises *"individuals and interactions over processes and tools"* [1]. However, the individual characteristics of the people working in the development teams, from the personality and cognitive perspective, have not been investigated to the extent they could offer improvements in continuous software engineering.

Human factors have been used in a plethora of cases of user-centered design techniques to provide effective personalisation, adaptation and improved user experiences [2]. Nonetheless, it has been reported that the main reason human factors still have failed to affect usability of systems on a large scale, is that the user-centred techniques have not been implemented in association with the software engineering community [3]. For the same reason, usability of the processes and methods in software engineering has not been given yet sufficient importance. For instance, the ways that tasks are designed and assigned to individuals do not explicitly consider human factors with regards to personality and individual differences in cognitive processing. A clear separation evidently exists between the developers and those that design the engineering processes, as the latter rarely take into consideration the individuals' backgrounds to specify the methods and techniques they use. Nevertheless, as professionals with different characteristics might need to work together in high performance teams, a better understanding of their individuality is very important. In addition, professionals perform tasks on various cognitive levels, collaborate in many constellations and have a mixture of experiences when interacting with development tools and methods. Therefore, the tools and methods they use could be adapted and personalised based on their cognitive processing abilities and styles. They could thus offer some improvement in efficiency and effectiveness of tasks execution in software engineering through bootstrapped functionalities.

The motivation of our work begins from the observation that software process productivity and efficiency critically depend on a variety of mainly human and social factors [4]. In continuous software engineering, where the development cycles are fast, an organisation needs to rapidly develop, deliver and learn in parallel. In order to rapidly respond to change and deal with the related uncertainty and complexity, it is important to accomplish a culture towards agile thinking, from the individuals, to teams and to the upper management levels. However, recent surveys indicate that the main reasons that agile is not easily adopted in practise are related to the difficulty of engineers to change their way of working, management resistance to change and lack of familiarity with agile concepts [5]. Earlier reports on "Peopleware" from the homonymous book [6] state that the most frequent cause of failure is complexity in "office politics", which includes public relations, interactions, staffing issues, management and customer disagreements, lack of motivation and high turnover. As such, the most critical problems have a social nature, rather than a technical one.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
RCoSE'14, June 3, 2014, Hyderabad, India.
Copyright 2014 ACM 978-1-4503-2856-2/14/06. $15.00.

To this end, this paper addresses the social aspects, starting from the individual level as current research clearly indicates that it is an essential part of an organisation's overall capability to accomplish continuous software engineering. From the individual perspective, a way to improve the performance of developers is to provide ways to improve themselves. By providing them tools to understand more about their individual qualities and the psychological aspects of other people, they can accept more responsibility in managing themselves and become more responsible for the project outcome [7]. Trust may be built, better forms of collaboration and communication flow of information could bridge together people, technology support and business intelligence to better accommodate the changing customer/employee needs and business workflows. Understanding individuals is critical in helping people thrive in organisations and in team constellations. Analysis of the personality aspects and cognitive processes can facilitate team formation and ultimately build the capability of organisations to perform efficiently, in an agile way and to complement the agile team orientation.

The overarching goal of this paper is to offer a position statement and proposal of future work towards personalised continuous software engineering. The contribution of the work is focused in providing methods to individuals to understand their cognitive and personality types and correlate differences and similarities while operating in high performance teams. As a remainder: Section 2 briefly describes the theoretical background. Section 3 explains the proposed approach of how human factors can influence continuous software engineering. Section 4 presents a case study and the preliminary results. Section 5 discusses the future research direction of the work, and Section 6 concludes the paper.

2. Personalised Software Engineering

Recent empirical studies on personalised software engineering have proposed that researchers should focus on the humans involved in the development, and one way to do this is to collect psychometric measurements [8]. However, a limited number of studies have focused on investigating relations between such measures and software engineering performance improvement, methods, tools and the quality of products.

In the related literature, human factors (e.g., experiences, opinions and perceptions) have been found to impact the effectiveness of software process improvement programs [9]. A series of investigations conducted on various people-related issues, highlighted the impact of agile methodology on individual well-being, personality on project outcomes, the level of conflict in different groups, the relationships between customers and programmers and in relation to software testing [9]. However, based on the analysis provided in a recent case study [10], a team abandoned its attempted transition to agile, and the main reasons were related to the lack of team/task familiarity, group cohesion and transactive memory (i.e., individuals' ability to encode beliefs in memory and beliefs about beliefs in the meta-memory [11]).

Another study investigating the relation of working memory and experience in the development of programming skills revealed a high correlation between the two [12]. Programmers with low working memory capacity have been found not able to continue to work on coding tasks for long periods (more than a few years). This was due to the fact that continuous learning is required from their side to stay updated with the language used for programming. Another finding of the study was that software developers with high levels of working memory are expected to be more productive than those with lower levels.

Based on the preliminary findings reported until now, we assume that the best way to approach a solution is to support the unique ways of thinking and perceiving information of engineers, improve their work performance and thereby the overall capability of organisations towards continuous software engineering by: (a) combining concepts of human factors with process improvement on the individual level, (b) forming teams based on the individual personalities of the people involved and provide personalised development environments based on the cognitive factors of the team members, and finally, (c) investigating how to ensure that the teams are harmonised with the organisation's development practices. The human factors approach is based on personality and cognitive processing, as described next.

2.1 Personality Factors

Personality is based on a set of individual characteristics that influence their cognitions, emotions, motivations, and behaviours in various situations. The *Big Five* personality traits are the five dimensions of personality that are typically used by psychologists to describe human personality. They include: *Openness* which reflects the degree of an individual's curiosity, creativity and preference for novelty, *Conscientiousness* which indicates an individual's tendency to show self-discipline and to act dutifully, *Extraversion* which indicates energy, positive emotions, assertiveness and sociability, *Agreeableness* reflecting the tendency to be compassionate and cooperative, and *Neuroticism* which indicates the tendency to experience unpleasant emotions easily (e.g., anger, anxiety, depression and vulnerability).

A number of researchers have proposed assessment tests that measure psychological preferences in how people perceive the world and make decisions, from the perspective of the Big Five personality traits. Examples include the Revised NEO Personality Inventory (NEO PI-R) which is a psychological personality inventory with 240-item measures, and its short version, the NEO Five Factor Inventory (NEO-FFI) containing 60 items [14]. Another popular personality assessment test is the Myers-Briggs Type Indicator (MBTI) [13] which is based on four personality types, Extraversion-Introversion, Sensing-Intuition, Thinking-Feeling, and Judging-Perceiving. Combinations of preferences form 16 psychological types. An individual's personality style based on the MBTI is commonly related with professional specialisation, and it is a popular instrument for describing personality type in the field of career and job counselling.

2.2 Cognitive Processing Factors

Theories of individual differences in human cognition aim to describe and explain how and why individuals differ in cognitive abilities and styles [15, 16, 17]. Various researchers attempted to explain the functioning of the human mind in terms of basic cognitive processes. They include for example the *speed of processing*, which refers to the maximum speed a given mental act may be efficiently executed [15]; *controlled attention*, which refers to cognitive processes to identify and concentrate on goal-relevant information and inhibit attention to irrelevant stimuli [15]; and *working memory capacity*, which is defined as the maximum amount of information the mind can efficiently activate during information processing [16]. Researchers have also focused on higher-level cognitive processes such as *cognitive styles*, which explain empirically observed differences in mental representation and processing of information [17].

Cognitive styles can be measured on two broad style dimensions [17]: (a) the *Verbal/Imager dimension* that refers to how individuals process information and indicates their preference for

representing information verbally (Verbals) or in mental pictures (Imagers), and (b) the *Wholist/Analyst dimension* that refers to how individuals organise information and indicates a preference in structuring information as a whole to get the big picture (Wholists) or structuring information in detail (Analysts).

2.3 Personality and Cognitive Factors in Software Engineering

In recent times, basic personality tests have been used to construct the profiles of software engineers [18]. Wynekoop and Waltz [19] contributed in identifying the personality traits of software developers. The six-fold model they proposed included characteristics and behaviours that distinguished top developers. Their characteristics included ability to abstract business problems, creativity, technical and business knowledge, work with and lead teams, analytical and logical thinking, high levels of self-motivation, dependability and organisation. In [20] the influence of effectiveness and efficiency to perform in a pre-determined role in a software process was related to individuals' behavioural competences or professional characteristics. The authors provided guidelines for including capabilities of people involved in the software process facilitating human resource management as regards to role performance and their capabilities in the tasks assigned. Even though their evaluation indicated some team performance improvement, problems raised related with the costs of the required training programs, availability (and willingness) of the staff to undergo the trainings, attitudes towards changing roles, management/union agreements in place and cultural resistance originating from the organisation itself.

Developers participated in a study and statistically significant correlations were found between personality and attitudes to software engineering processes and tools [8]. High levels of conscientiousness were correlated with preferring to work alone, low need to change the current working processes, preference to working on non-technical tasks, rather than technical ones, and thinking that software engineering research is not that important, were a few of the observations. The authors also discussed in their work the merits of the IPIP-NEO test [21, 22] and why in many cases it can be preferred over the popular MBTI [13].

Recently, cognitive factors and other internal individual differences, other than personalities, have come into focus in software engineering [23]. To the best of our knowledge, the work in [23] and our own are one of the few ones combining both personalities and cognitive processing of developers. Feldt et al. [23] link personality, decision-making and performance in software engineering, while we focus also on cognitive abilities.

3. PROPOSED APPROACH

A personalised approach of integrating human factors in continuous software engineering combines the concepts of human factors and software engineering and investigates the current states of team forming and adaptation of tools and methods. The human factors that we investigate in this work include personality factors and cognitive processing characteristics. They were selected because we assume that they cover both human aspects of logical and emotional thinking. In this respect, we propose that considering human factors in continuous software engineering can possibly help in increasing the levels of satisfaction in development teams. The hypothesis is that efficient teams can be formed if this knowledge is made available to the team members. In addition, it can allow strengthening the feeling of responsibility, increase satisfaction, influence positively continuous development and self-organisation in agile. For instance, pairing people in pair programming would require finding a spectrum of people and skills, i.e., people producing a high number of bits code and people critically thinking, analysing and testing parts of the code (Analytic cognitive style) and also, people looking at the overall picture and architecture of the system (Wholistic cognitive style). Also, acknowledging compatible or incompatible personality types in high performing teams could be proven particularly useful in team formation.

Agile software development relies on the talent, skill and resourcefulness of humans, rather than the process itself [24]. By aligning human factors, i.e., personality traits and cognitive thinking, with team interactions is a way to influence continuous and agile development. Another way is to adapt agile methods within the context of operation, to satisfy the unique preferences and types of self-organised teams. The effect of human factors in this realm is proposed in two approaches: (a) *Team Formation*, improving ways of forming teams based on their collective personality traits, and (b) *Tools and Methods Adaptation*, by providing personalised tools and methods to software engineers based on their individual differences in cognitive processing.

3.1 Team Formation

According to Tuckman [25], high performing teams develop through four main phases of team formation: forming, storming, norming, and performing. From the perspective of continuous software engineering, and its underlying emphasis on agile development practices, the last phase of team formation (performing) corresponds to the ways self-organised agile teams work. However, one of the greatest challenges for teams is progressing through the phases of team development to form an effective team, taking into account the particular context and also their underlying software capabilities [26]. We assume that by merging this process with knowledge about personality traits and their impact on the team and its collective performance could potentially accelerate team forming.

The results reported in [8] revealed several strong associations between personality aspects and software engineering related views, attitudes and work preferences. Given that personality metrics provide useful insights about the individuals, they also offer a useful tool to assess the power of personality and impact on a team's performance. More specifically, using this knowledge to analyse problems throughout the phases of team formation and respond accordingly, can provide a faster way to form effective teams. Ultimately, the effects of capable teams are reflected in an organisation's overall ability to high performance [26]. From a bottom-up perspective, this approach offers a crucial part of the solution to bring software engineering a step further to being a holistic continuous process.

Additionally, there are many outstanding issues related with the effective forming of teams. For instance, related research in software engineering addresses the aspect of matching personalities in development teams. Rutherfoord [27] reports the results of using the Kiersey Temperament Sorter [28] in an experiment of forming teams in software engineering projects within an educational environment, and concludes that the strongest teams are those that have heterogeneous personalities. The author also mentions that using a personality inventory could help establish not only heterogeneous groups, but also create groups that understand and appreciate the strengths and weaknesses of all team members. More interestingly, existing research shows that there are many different personality and motivational models and theories, and each one is offering a

different perspective. However, there is still not a clear agreement on which model is most appropriate and should be adopted. There are many dimensions to be addressed yet, such as the ethics of forming teams and recruiting people based on personal traits. Even though studies in other disciplines address such issues, they are still limited in the particular software engineering domain.

3.2 Tools and Methods Adaptation

An adaptation of the tools and methods used in continuous software engineering could offer increase in productivity and efficiency. Applications could include personalisation of specific tools and methods used by software engineers, like e-learning platforms (or online documentation), development/testing environments and collaboration platforms. Previous research has shown for instance that the development processes, methods and tools used by individuals may be more efficient if their individual differences are taken into consideration for personalisation [8].

Similarly, in the context of continuous software engineering, we propose that adapting the tools and methods based on the developers' working memory capacity could minimise their cognitive load. By offering for example personalised code views, indexing of help and navigation support can alleviate developers' cognitive burden and allow for active decision making (i.e., selecting between several solutions the best alternative). Personalised tools and methods for specific engineering roles, such as programmers and architects, can facilitate the process of coordinating important data and improve decision making. In addition, they can improve performance by facilitating the diagnosis of organisational and software problems (such as code smells and defects), as well as, their possible causes that would not be easily visible otherwise.

Various forms of adaptations of tools and methods could be appropriate in particular contexts. They are generally classified below as *adaptive presentation* and *adaptive navigation support* [2] and related research work is provided as examples.

Adaptive presentation of content relates to the way information (links, text, graphics, etc.) is presented in digital environments. The information, for example in the case of developers, can represent fragments of information adapted for developers that have a specific state (level) of knowledge. They can provide the appropriate missing prerequisite knowledge, details, or comparison with a previously known concept to simplify code authorship or locate help items in documentation materials. Techniques that can be used to provide adaptive presentation include among others: (a) inserting/removing fragments relevant to the developer's tasks, (b) expanding/collapsing content fragments (e.g., providing/hiding additional explanations to novice and expert developers accordingly), (c) altering (simplifying/advancing) content fragments, and (iv) sorting content fragments (e.g., providing an example before the class definition to novice developers).

In related work, the authors in [29] proposed an environment to offer software developers views of unified code fragments with additional information about the code (e.g., parameters used). They also enabled/disabled random navigations - backwards and forwards - if they were considered to hinder for example developers that were Wholists. Such restrictions minimised the risk of users losing orientation in the environment. Wholists were also given extra guidance to view code fragments and Imagers were given diagrammatical representations (i.e., code bubbles) to increase efficiency in process information during coding.

Adaptive navigation support relates to the adaptation of tools used for navigation either in development, collaboration or knowledge-base environments. Such adaptation supports developers' navigation based on their goals, preferences and knowledge. They support developers to filter important parts of their code, like parts not passing the tests. Adaptive navigation support can be achieved by: (a) showing information that is mostly relevant to developers in a hierarchy, (b) hiding, removing and/or disabling information to restrict navigation to irrelevant content, and (c) augmenting content with additional information, with some form of annotation, adding more to the navigation options.

In related work [30], the authors proposed adaptive navigation support by providing tools for viewing additional explanations of particular concepts in an e-learning environment for programming and algorithms. They used cognitive styles as determinant factors for presenting the explanations/suggestions to Wholists in the form of tooltips and to Analysts in the form of pop-up windows to assist their learning and comprehension process.

The study that follows aimed to elicit the need and motivation of the work that we propose, considering personalities and identifying individual characteristics of members in high performing teams in agile environments.

4. CASE STUDY

4.1 Sampling and Procedure

The case study was conducted in the development environment of two companies in Sweden and Greece. It aimed to discover if human factors can be useful for continuous software engineering. The main issues investigated include if there is a motivation in employing cognitive thinking and processing, personalities and compatibility in forming high performing teams and adapting tools and methods based on individuality.

Two teams participated consisting of developers aged on average 35 years old. The teams have been formed during the past 2-3 years and all individuals have more than 3 and 6 years on average of work experience. The Swedish team uses Linux as a platform and programs in C, C++ and Java, while the Greek team uses J2EE. The tools they typically use for programming range from Eclipse, to Emacs and Vi editors in the Swedish team and NetBeans, IntelliJ and Eclipse in the Greek team. The satisfaction questionnaire, personality and cognitive psychometric tests utilised for the purpose of the case study can be found online[1].

4.2 Discussion of Results

An analysis based on the *Big Five* dimensions of personalities was made using the IPIP-NEO test [21, 22]. Both teams in their majority included members scoring high on the Agreeableness (A) characteristic. Some of the team members in the Swedish team scored high on the Extraversion (E) and Neuroticism (N) characteristics. Most members in the Greek team scored high on the Conscientiousness (C) characteristic, thinking hard about their responsibility in taking actions and decisions, and also expressed high anxiety levels. Table 1 lists the values for the personality dimensions of Openness (O), Conscientiousness (C), Extraversion (E), Agreeableness (A) and Neuroticism (N) for the Greek team members. Relating personalities and collaboration, in the Swedish team, some of the members that expressed to collaborate well together in the team environment, shared not only many similar personality traits, but also expressed increased satisfaction levels

[1] http://www8.cs.ucy.ac.cy/projects/personaweb/um/study.php

of their collaborations within the team (i.e., in responses like: "*I feel happy collaborating with my team-members*"(Q1 of the satisfaction questionnaire)).

The collective team responses showed that both teams are feeling highly motivated in their work. However, the Swedish team replies reflected a state of "command-and-control" environment in that they are self-organised only to the extent that they can select their tasks within their teams. In their perspective, self-organisation is being implemented naturally, only in some occasions. Their working environment restricts their possibility to select team mates because it is typically the managers' responsibility to allocate resources. However, they expressed strong opinion in that if they would work with someone they get along they could improve their self-organisation ability as a team.

Table 1. Greek team personality types

	G1	G2	G3	G4	G5	G6	G7	G8
O	9	40	17	94	15	96	21	91
C	56	1	22	73	79	66	46	86
E	64	9	40	27	52	71	38	63
A	35	49	60	71	35	86	77	56
N	21	85	37	24	8	43	45	14

Overall, the responses obtained in the satisfaction questionnaire indicated that the organisation and productivity of teams could be improved if team members were assigned suitable tasks, based on their personality and they were positive in the idea of tool adaptations based on their cognitive ability, indicating some proof in our motivation for conducting this research.

As the members of the teams were involved in various projects during the period which they constituted a team, it was interesting to analyse sub-groups of the teams that were more productive during their collaborations. In specific project cases (marked as Pr. below), the most valuable player, 'smooth' recent project and valuable pairs were indicated by the project manager in an interview. For instance, in the Greek team, the high performing pairs of G3-G6, G4-G1, G7-G4, G8-G4, G8-G6 and G8-G1 were identified for 6 different projects (Pr. 1-6). An observation made is that the individuals in the highest performing pairs have had a correlation in their personalities, in all cases except in Pr. 3 (G7-G4). They also matched most of the times in terms of the Neuroticism (N) characteristic, being in quite low value. This means that there is an indication that individuals with correlated personality types might perform better as a pair than non-correlated personalities. It is also very interesting to analyse smooth projects such as Pr. 5 and their high performing pairs.

Even though some interesting observations can be made from this case study, the number of individuals participated is too small to reach to general conclusions. The relationship between human factors and software development still needs to be further studied. Human factors can be however used to better understand individuals and form efficient development teams which are considered complex to balance. Teams typically will comprise a mix of individuals with diverse characteristics, i.e., personalities, capabilities, commitments, experiences, motivations and ambitions. Their efforts and attitudes will need to be aligned and since they depend on human factors, understanding them better in terms of interactions, collaborations, ways of thinking and working can facilitate team formation and ultimately productivity.

5. FUTURE RESEARCH DIRECTION

A future direction is to expand to something that many organisations struggle with today, which is, taking Agile beyond IT departments, and creating suitable holistic structures supporting continuous software engineering. In this paper, we argue that not only suitable governance models are needed on the top organisational levels, but also models considering the individuals' personal traits to accommodate the objective of holistic continuous software engineering. Although there is a lot of interest in agile at the enterprise level, as recent research shows, there are no clear compelling models for how this could work. Examples such as SAFe [31] and Stairway to Heaven [32] provide guidance when it comes to large development projects and programs. However, they do not provide guidance on the individual level, such as team formation models considering personal traits and promoting a compatible organisational culture throughout all organisational levels. Another future direction includes expansion of the current research and working in collaboration with development teams to improve their efficiency. As prior research within other disciplines is rich, for example in health care and psychology, a reasonable extension could be to analyse their studies, learn more about individuals, team, various implications and challenges that can be used to accelerate productivity and the team formation process. Also, we agree with the need of methodology and empirical basis to relate personality and cognitive factors in software engineering research and practise, as identified in [8], and we plan in the future towards addressing some of the challenges and open issues in this area.

6. CONCLUSIONS

This research associated efficiency and performance of development teams in the context of continuous software development with human factors. We suggested that using human factors to design teams, tools and methods can potentially improve productivity and the overall satisfaction in the ways of working of software engineers. A case study was conducted and provided indications that this suggestion is true in the particular context. Individualised continuous software engineering can potentially serve efficiently the individual needs and preferences of the teams and the people involved in the development process. This work is still at an early stage, but we aim at continuing our efforts in showing at a larger scale the effects as recognised in the case study. The overarching target is to contribute to improving the communication, cohesion, efficiency and performance of teams. Nevertheless, to reach to more concrete results the relation of individual differences and capability in continuous software engineering warrants further study.

7. REFERENCES

[1] Manifesto for Agile Software Development, http://agilemanifesto.org/, Accessed January 2014.

[2] Brusilovsky, P., Kobsa, A., and Nejdl, W. 2007. *The Adaptive Web: Methods and Strategies of Web Personalization*. Springer, Heidelberg.

[3] Seffah, A., and Metzker, E. 2004. The Obstacles and Myths of Usability and Software Engineering. Commun. ACM 47, 12, 71–76.

[4] Boehm, B.W., Abts, C., Brown, W.A., Chulani, S., Clark, B.K., Horowitz, E., Madachy, R., Reifer, D.J., and Steece, B. 2000. *Software Cost Estimation with COCOMO II*. Prentice Hall, Upper Saddle River, NJ.

[5] Papatheocharous, E., and Andreou, A.S. 2013. Evidence of Agile Adoption in Software Organizations: An Empirical Survey. In *Systems, Software and Services Process Improvement*, F. McCaffery, R.V. O'Connor, and R. Messnarz, Eds. Communications in Computer and Information Science 364. Springer, Berlin Heidelberg, 237–246.

[6] DeMarco, T., and Lister, T.R. 1999. *Peopleware Productive Projects and Teams*. Addison-Wesley, Upper Saddle River, NJ.

[7] Gómez, M. N., and Acuña, S.T. 2013. A replicated quasi-experimental study on the influence of personality and team climate in software development, *Empir. Softw. Eng*. DOI= 10.1007/s10664-013-9265-9.

[8] Feldt, R., Torkar, R., Angelis, L., and Samuelsson, M. 2008. Towards individualized software engineering: Empirical Studies Should Collect Psychometrics. In *Proceedings of the 2008 International Workshop on Cooperative and Human Aspects of Software Engineering* (Leipzig, Germany, May10-18, 2008). CHASE '08. ACM, New York, NY, 49–52.

[9] Viana, D., Conte, T., Vilela, D., de Souza, C.R.B., Santos, G., and Prikladnicki, R. 2012. The Influence of Human Aspects on Software Process Improvement: Qualitative Research Findings and Comparison to Previous Studies. In *Proceedings of the Evaluation & Assessment in Software Engineering* (Ciudad Real, Spain, May 14-15, 2012). EASE 2012, 121-125.

[10] Ralph, P., and Shportun, P. 2013. Scrum Abandonment in Distributed Teams: A Revelatory Case. In *Proceedings of the Pacific Asia Conference on Information Systems* (Jeju Island, Korea, June 18-22, 2013). PACIS 2013.

[11] Wegner, D. M. 1987. Transactive Memory: A Contemporary Analysis of the Group Mind, In *Theories of Group Behavior*, B. Mullen, and G. R. Goethals, Eds. Springer, New York, 185–208.

[12] Bergersen, G. R., and Gustafsson, J.-E. 2011. Programming Skill, Knowledge and Working Memory Among Professional Software Developers from an Investment Theory Perspective. *J. Individual Differences* 32 (4), 201-209.

[13] Myers Briggs, I., and Myers, P.B. 1995. *Gifts Differing: Understanding Personality Type*. Davies-Black Publishing, Mountain View, CA.

[14] Costa, P.T.Jr., and McCrae, R.R. 1992. *NEO PI-R Professional Manual*. Psychological Assessment Resources Inc.

[15] MacLeod, C.M. 1991. Half a Century of Research on the Stroop Effect: An Integrative Review. 1991. *J. Psychological Bulletin* 109, 163-203.

[16] Baddeley, A. Working Memory: Theories, Models, and Controversies. 2012. *J. Annual Review of Psychology* 63, 1-29.

[17] Riding, R., Cheema, I. 1991. Cognitive styles – An overview and integration. *Educational Psychology* 11 (3-4), 193-215.

[18] Moore E. 1991. Personality Characteristics of Information Systems Professionals. *In Proceedings of the 28th Annual SIGCPR Conference on Information Systems in Personnel Interface.* (Athens, Georgia, April 8-9, 1991). SIGCPR '91. ACM, New York, NY, 140-155.

[19] Wynekoop, J, and Walz D. 2000. Investigating Traits of Top Performing Software Developers. *Inform Technol People* 13 (3), 186–195.

[20] Acuña, S.T., and Juristo, N. (2004). Assigning people to roles in software projects. *Software: Practice and Experience* 34 (7), 675–696.

[21] Goldberg, L. R. 1999. A Broad-Bandwidth, Public-Domain, Personality Inventory Measuring the Lower-level Facets of Several Five-factor Models. *Personality Psychology in Europe* 7, 7–28.

[22] Goldberg, L.R. Johnson, J.A., and Eber, H.W. 2006. The International Personality Item Pool and the Future of Public-Domain Personality Measures. *Research in Personality*, 40 (1), 84–97.

[23] Feldt, R., Angelis, L., Torkar, R., and Samuelsson, M. 2010. Links Between the Personalities, Views and Attitudes of Software Engineers. *Inf. Softw. Technol.*, 52 (6), 611-624.

[24] Cockburn, A., and Highsmith, J. 2001. Agile Software Development, the People Factor. *Computer* 34 (11), 131–133.

[25] Tuckman, B. 1965. Developmental Sequence in Small Groups, J. *Psychological Bulletin* 63 (6), 384–99.

[26] Kettunen, P. 2014. Directing High-Performing Software Teams: Proposal of a Capability-Based Assessment Instrument Approach. In *Proceedings of International Conference on Software Quality Days* (Vienna, Austria, January 14-16). SWQD 2014.

[27] Rutherfoord, R. 2001. Using Personality Inventories to Help Form Teams for Software Engineering Class Projects. In *Proceedings of the 6th Annual Conference on Innovation and Technology in Computer Science Education* (Canterbury, UK, June 25-27, 2001). ITiCSE '01. ACM, New York, NY, 73-76.

[28] Keirsey, D., and Bates, M. 1984. *Please Understand Me*. Prometheus Book Company, Del Mar, CA.

[29] Bragdon, A., Zeleznik, R., Reiss, S.P., Karumuri, S., Cheung, W., Kaplan, J., Coleman, C., Adeputra, F., and LaViola, J.J. 2010. Code bubbles: a working set-based interface for code understanding and maintenance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Atlanta, USA, April 10-15, 2010) CHI '10. ACM, New York, NY, 2503-2512.

[30] Germanakos P., Tsianos N., Lekkas Z., Mourlas C., and Samaras G. 2008. Realizing Comprehensive User Profile as the Core Element of Adaptive and Personalized Communication Environments and Systems. *Comput. J.* 52, 7 (October 2009), 749-770.

[31] Leffingwell, D. 2007. *Scaling Software Agility: Best Practices for Large Enterprises*. Pearson Education, Boston, MA.

[32] Olsson, H., Alahyari, H., and Bosch, J. 2012. Climbing the "Stairway to Heaven" - A Mulitiple-Case Study Exploring Barriers in the Transition from Agile Development towards Continuous Deployment of Software. In *Proceedings of EUROMICRO Conference Software Engineering and Advanced Applications* (Cesme, Izmir, Turkey, September 5-8, 2012). SEAA 2012.