Available online at www.worldscientificnews.com

World Scientific News

WSN 81(2) (2017) 64-75 EISSN 2392-2192

Development of software engineers' competences in context of formal education

Michał Adamczyk

AGH University of Science and Technology, Faculty of Management, Department of Organisational Management, Human Resources Management and Economic Law, 10 Gramatyka Str., 30-067 Cracow, Poland

E-mail address: michal.adamczyk@agh.edu.pl

ABSTRACT

The purpose of this paper is to discus key competences of knowledge workers on the example of software engineers (developers). First part presents an overview of the literature on both the desired competences of programmers and methods of their development. Second section presents results of quantitative research based on on-line survey conducted among IT professionals and academic teachers. Last part contains conclusions and recommendations for formal education.

***Keywords*:** Soft skills, technical skills, competences, software developers, programmers

1. INTRODUCTION

Development of information technology (IT) is an inseparable element of the knowledge economy. IT specialists are considered to be good example of knowledge workers, whose professional activity relay mainly on their individual competences. In this context it is safe to assume, that intellectual capital of students graduating computer science studies is an important factor influencing development of domestic economy. This aspect is even more important for policy makers of regions like Małopolska (district in south Poland), where Information and Communication Technologies (ICT) were declared as one of seven priorities for regional strategy of development.

The purpose of this paper is to identify key competences of software developers and compare them with formal academic education. Author believes that conducted research will be important guidelines for academics and policy makers. The next two sub-sections of this paper will present overview of the literature on both the desired competencies of programmers and methods of their development. Next section presents results of quantitative research conducted among IT professionals (mostly form Małopolska) and academic teachers (form leading polish technical universities). Last section contains conclusions, recommendations for formal educators and some proposals for further studies.

1. 1. Required competences

The basic tool of software developer is obviously a computer with special software. This method of work is called computer aided software engineering (CASE). It requires a set of tools for helping software developers to create their software more efficient. Coupe and Onodu confirmed positive impact of CASE on software quality and productivity in various development life-cycle stages. According to software developers, CASE also increases reliability of created software, what leads to reduction of corrective maintenance and increases general business performance. There are also evidence suggesting that insufficient experience reduces benefits of using CASE [4].

A lot of research and development effort is invested in upgrading technical environments and creating new tools for software development. However despite of any technical solution, human factor is crucial to project success and therefore it is safe to assume, that individual competences of programmers are the most important ‘tools’ for efficient software engineering [4,6].

Turley and Bieman described a two-phase study on software developers. At first they conducted in-depth interviews with 10 programmers recognized as ‘exceptional’ and 10 considered ‘non-exceptional’. Myers-Briggs Type Indicator and Critical Incident Interview was employed to identify 38 essential competencies of software engineers. In second phase surveys were conducted among 129 software engineers to verify which of identified competences were statistically relevant [14].

According to Turley and Bieman MBTI test revealed no significant differences between ‘exceptional’ and ‘non-exceptional’ developers, what could mean that personally type is not a good predictor of performance. However researchers identified and described 5 characteristics, that makes software developer ‘exceptional’ [14]:

1) They are focused on helping other programmers to solve their problems and also spends much time on teaching them how to improve their skills.
2) They pro-actively cooperate with top management and influence business decisions concerning project.
3) They demonstrate strong convictions and ‘acts in accordance with them, even when they are counter to specific management direction’.
4) They are highly skilled and work comfortably with many design patterns and coding techniques, choosing the right tools for a specific task.
5) They see ‘big picture’ and do not focus their attention on irrelevant details. They tend to have full knowledge about work of other team members to fully understand goals of project.

Turley and Bieman identified that ‘use of prototypes’ is a differential characteristic for exceptional and non-exceptional software developers, although they could not fully verify that observation in second stage of their study [14]. Their results are however consistent with those presented in the next section of this paper.

Based on quantitative research, Hung-Lian, Lee and Koh identified education gaps between required and achieved competences in areas of interpersonal communication and behavior skills, critical and creative thinking, methods of architecture design, personal motivation, self-organization [7].

Lee and Han conducted research based on the review of 837 job offers for position of ‘entry level’ IT specialist posted on websites of corporation ranked on Forbes 500 list. Software developers or broadly speaking IT specialists are considered to be rather business partners than technicians. Over 75% of ads mentioned business knowledge. Authors reported that also ‘functional knowledge including human resources, management, marketing, finance, etc.’ was mentioned in almost 70% of job ads. In about one third of cases flexibility and addictiveness is required from potential candidates, also ‘a quarter of the ads contained phrases related to being customer-oriented’. Authors concludes, that ‘overall, skills related to development, software, social skills and business were highly required for programmers/ analysts by large corporations’ [9].

Based on survey conducted by Cappel among IT employers, the gaps between actual and expected competences are higher for non-technical than technical. Non-technical skills are often rated higher than technical skills. It can be explained by the fact, that technical skills are specific for certain IT jobs, while competences like team communication, problem solving, learning ability, etc. apply to most IT-related jobs [2].

Other researchers reported that important attributes of software developers are: pro-activeness, flexibility and adaptability, fully documenting work and sharing knowledge with the team. Crucial things are pro-activeness in problem solving, communication skills and ‘seeing the big picture’. In today’s complex IT systems, tech skills are not enough, however according to respondents, technical competence was seen as the crucial factor in software performance [6].

Bailey and Mitchell identified even 85 competences important to software developer. They divided them in three groups: technical, soft skills and business concepts. Later, based on web surveys, 23 of thouse competences were considered to be highly important: 13 of them were technical skills and 10 soft skills. Three competences identified as most important were technical: ability to modify, write and debug source code. Next three were soft skills: ability to listen others, problem-solving and team work skills [1].

Sterling and Brinthaupt distinguished desired competences of individual programmers and those working in groups: ‘A successful individual programmer has strong technical and cognitive and problem-solving skills, and is conscientious and creative’, while desired team programmer ‘is skilled at interpersonal interaction and cooperation, shows self-confidence and maturity, and is creative, analytical, and conscientious’. However authors admit, that in business practice this division between solo and team-workers is impossible, because in most cases IT professionals has to perform some tasks in group and some individually [13].

Ss shown above, studies on IT competences provide conflicting evidence about importance of technical and soft skills. Most studies conducted with desk research method (job ads analysis) shows that technical competences are most important factor while applying

for job in IT. On the other hand more empirical research often emphasizes importance of soft skills [11].

Litecky, Arnett and Prabhakar tried to explain this paradox with usage of Image Theory. According to that behavioral decision-making concept, the process of hiring people has two stages. The first is about screening out bad candidates, when the second stage is about choosing the best options. Researchers believe, that while hiring IT specialist, the technical skills are analyzed at first stage of recruitment process and candidates who do not match desired level of competences are rejected. In second stage, while actual hiring decision is being made, mostly non-technical competences are taken into account [11].

1. 2. Competences development

According to study conducted by Cappel the most important competence for all IT professionals is ability to learn [2]. This is a characteristic feature of modern economy, where everything is dynamic and once acquired competence must be constantly renewed. This is not a great challenge for software developers, as they own a strong need for personal development and increasing their competence level [6]. Analysis of empirical data suggests that employee turnover can have positive impact on IT knowledge [5].

On the other hand, however Turley and Bieman showed that experience has no direct correlations with performance [14]. Qualitative studies conducted by Coleman and O'Connor according to Grounded Theory method on 21 IT start-ups revealed, that software developers tend to use engineering method which they learned in previous company. In many cases they use method that are not necessary best for their current organization, but were created for their previous one. This shows how much developer relay on their own experience [3].

This could lead to a conclusion, that although, as Cappel stated 'there is no substitute for job experience' [2], learning process of software developer is a bit more complicated than just gaining experience. Developers often change their job in search for new challenges, as they want to learn new technologies and work in different environments to develop new skills.

Lee et al. discovered an interesting fact, that more experience IT professionals have higher non-technical competences, while their younger colleagues have higher technical competences. This is explained by the fact, that soft skills are gained through interaction in workplace, so the longer they work, the better they are skilled. On the other hand technological development makes technical skills obsolete quickly [10].

Experience is undoubtedly one the most important sources of knowledge for software developers. Another interesting factor is formal education. Most of job ads analyzed by Lee and Han required higher education diploma (bachelor's degree or higher). However possessing a formal industry certificate is not a crucial factor while applying for jab as IT specialist [9].

Many studies have been devoted to aspects of formal education. Iqbal and Harsh propose more complex approach for teaching programming. They claim that academic curriculum mostly focus on teaching syntax and semantics of programming language, not seeing the 'bigger picture'. They believe that teaching materials should be more focused on planning and monitoring results and understanding the whole complexity of software engineering, rather than just coding, as novice programmers often start their work by just writing the code without any planning and pay little attention to testing [8].

Cappel asked employers for their expectations towards entry-level job skills. He also made some advices for IT teaching programs. Obviously programming abilities remains a crucial competence. Interestingly, according to employers, it doesn't make much difference which programming language is being thought. They sees 'language courses as means to teach programming concepts, even though the actual language a student might use after graduation could be different' [2].

Sterling and Brinthaupt believes that programming teachers should pay more attention to cognitive, problem-solving and other non-technical skills'[13]. Hung-Lian, Lee and Koh suggests introducing team projects during programming courses, so students could develop interpersonal and organizational skills [7]. Cappel suggested that 'there is the greatest room for improvement in new IT graduates regarding non-technical skills' [2].

Another, frequently discussed aspect in the context of IT competences development is so-called 'pair programming'. Structured experiment was conducted on University of Utah, where senior software engineering students were divided into two groups both composed of the same mix of high, average, and low performers, as determined by their grade point averages. All of them attended same classes and receives the same instruction, however first group was instructed to work individually, when second group worked in pairs. During six weeks students received four assignments [15].

Initially programmers were skeptical about this way of working, as they were used to work individually. They needed some time to adjust to this method. First assignment took about 60% more programmer hours to complete, but after some adjustments, it was reduced to 15%. It has been statistically proven, that pair-programming produced higher quality code.[15] Similar test was conducted in Federal College of Education in Nigeria on group of 68 students. Results also confirm that increased interaction between developers produces better results [12].

2. RESEARCH CHARACTERISTIC AND RESULTS

The purpose of this research was to identify key competences of IT professionals and impact of formal education on their performance. Three research questions were stated:

1) What are the key competences of IT professionals that influence their performance?
2) Is formal education an important source of knowledge for IT professionals?
3) Is there significant difference in perception of IT competences between academic teachers and professionals?

Quantitative approach was employed to answer those questions. Data was gathered using Computer Aided Web Interview (CAWI). Participants were provided with a special hyper link leading to an on-line survey. Acquired data were then analyzed with the support of IBM SPSS software. Two statistical tools were used to verify relationships between specific variables: Student's t-test and Kruskal–Wallis H test.

2. 1. Pilot study

First part of research was pilot study conducted on 133 IT-graduates. Participants were asked to fill on-line survey containing closed and open questions. They were asked to rank importance of predefined competences (based on researches mentioned in previous section) but they also had an opportunity to add skills, which they found important and other comments regarding IT-competences in their every day work. Based on their responses it was quite hard to build one competence profile for all IT specialists, due to variety of work they were involved in. Figure 1 presents number of respondents declaring their affiliation to different fields from software development to technical support and business related activates. Number of answers do not sum to 133 as participants were allowed to check more than one field. Only 41 respondents declared one field of activity, same number declared their involvement in two fields, 30 were involved in three fields and 16 in four or more.

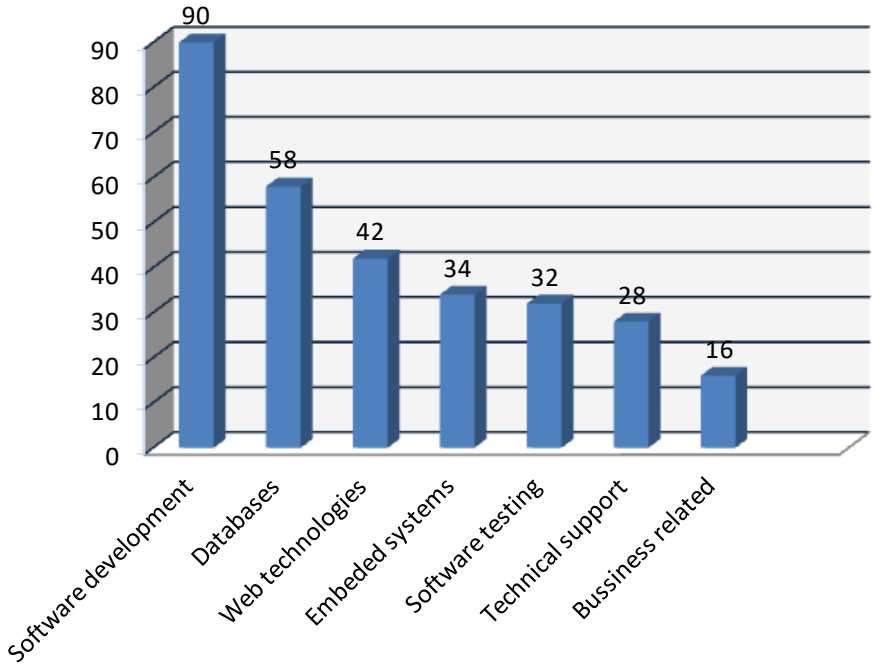

Figure 1. Number of respondents from various IT sectors in pilot survey. (Own research)

Most of IT-graduates participating in pilot study were somehow involved in software development, therefore further research was limited to this sub-group of IT professionals. Based on their responses (and literature review), 13 key competences of software developers were introduced. Eight of them could be qualified as technical skills:

1) *Knowledge of algorithms*. Computer science knows many ready to use general algorithms, which can be used to solve problems like data compression, elements sorting, finding patterns etc. Software developer can use them to save time and effort. In most cases well developed and tested algorithms described in literature produces better outcome than created ad-hoc by developers.

2) *Knowledge of design pattern*. In software engineering design patterns can be explained as general conceptions for designing software architecture. One the most commonly used design patterns is so called MVC (Model-View-Controller), which is used to organize software in three sub-systems: logical operations (model), data presentation (view) and data acquisition (controller).
3) *Knowledge of libraries*. Libraries are ready to use parts of code, that can be used to created a specific functionality. For example, very popular library called OpenGL enables coders to easily create and manage 3D graphics.
4) *Knowledge of versioning systems*. This are special tools for managing source code (or any other texts) while working in a team. Versioning system allows coders to 'reconcile' a common text version while it is being created by multiple authors.
5) *Knowledge of IDE*. Integrated Development Environment provides a set of tools for supporting software creation. IDE usually consists of special text editor, compiler and other tools for designing and testing software.
6) *Ability to create portable code*. Well written source code can be re-used in other projects. Experienced programmer knows about that and creates his work the way in can be used in the future by him or any other developer.
7) *Ability to test software*. While working on a complex software, one part of it can influence others. In order to avoid the need to continually repair emerging bugs, developers use so-called "tests", special sub-programs that constantly checks the correctness of each component.
8) *Ability to write clean code*. Coders can't work like novel writers, who have their own style. They must use a common way of creating code, naming specific elements and using special comment, which makes it possible for other developers to fully understand their work.

Five of identified key skills could be considered as 'soft':

1) *Ability to organize work.* It is important, especially when management cannot fully control the work of IT professionals and in many cases solo programmer must organize his work on his own.
2) *Ability to acquire knowledge.* Rapid development of IT industry enforces developers to work with unknown technologies. On the other hand there are many source of information. Good software developer must be able to quickly acquire and use new knowledge.
3) *Customer orientation abilities.* Programmers must remember that they create software for clients. Developer who poses this competence focuses on future user needs on every step of creating software.
4) *Ability to learn from others.* The determinant of this competence is the frequency with which a given programmer refers to other IT specialists when a technical problem occurs.
5) *Ability to share knowledge.* In this case, the distinguishing factor was the frequency of technical assistance to other programmers.

2. 2. Empirical research

Thirteen competences identified in pilot study were used to create two separate surveys: one for professionally active software developers and second for academic teachers of computer science. Both groups were asked to rank importance of 13 competences using 5-step Likert's scale. First survey was filled by 161 respondents, which were identified and ask to participate in research through LinkedIn portal. Second was send by e-mail to 17 departments of computer science at top polish technical universities and filled by 62 participants. Table 1 presents differences in responses of both groups. T-test was employed to determine statistical significance between them.

Table 1. Importance of specific competences according to software developers and academic teachers. (Own research)

Name of competence	Level of importance		Relative difference	Statistical significance
	According to software developers	According to academic teachers		
Knowledge of algorithms	3,58	4,32	18,6%	,000
Knowledge of versioning systems	4,51	4,26	6,3%	,044
Ability to create portable code	3,61	3,85	6,2%	,097
Ability to share knowledge	4,09	3,87	5,4%	,101
Ability to organize work	4,54	4,34	5,0%	,077
Ability to learn from others	4,24	4,05	4,8%	,106
Knowledge of design pattern	3,98	4,15	4,3%	,152
Ability to acquire knowledge	4,71	4,55	4,0%	,117
Knowledge of IDE	4,02	3,90	3,0%	,413
Ability to write clean code	4,67	4,60	1,9%	,393
Ability to test software	3,96	4,03	1,7%	,589
Knowledge of libraries	4,27	4,31	0,8%	,772
Customer orientation abilities	4,25	4,26	0,2%	,944

As we can see from Table 1, the most significant difference in opinions between teachers and software developers considers knowledge of algorithms and versioning systems. Academics tends to put more attention to more theoretical aspects, when software developers focuses more on code management.

Table 2. The impact of competencies on project status. (Own research).

Name of competence	Statistical significance of project status		
	Success	Challenged	Failure
Knowledge of algorithms	,775	,156	,409
Knowledge of versioning systems	,031	,112	,854
Ability to create portable code	,215	,477	,494
Ability to share knowledge	,164	,096	,160
Ability to organize work	,000	,000	,162
Ability to learn from others	,377	,820	,676
Knowledge of design pattern	,015	,223	,199
Ability to acquire knowledge	,321	,198	,645
Knowledge of IDE	,183	,429	,216
Ability to write clean code	,307	,241	,254
Ability to test software	,516	,832	,327
Knowledge of libraries	,323	,786	,597
Customer orientation abilities	,108	,239	,043

Second part of study was to determine which competences have an actual impact on software performance. Participants (161 software developers) were asked to mark on 5-step Likert's scale how often they demonstrate a specific competence during their ever-day work. All competences were operationalised with corresponding behavioral determinants. For example programmers were asked: 'How often do you cover your code with tests?' or 'How often do you use design patterns (e.g. MVC, SOA, P2P)?' Respondents were also asked how

much percentage of projects they are working on ends with: success, failure or how many of them are challenged (in terms of budget, schedule or functionality).

Table 2 presents relationship (H-test statistical significance) between frequency of specific behavior and project status. Factors having most significant impact on IT projects are: ability to organize work, knowledge of designs patterns and versioning systems, ability to orient pro customer and ability to share knowledge with other project members.

Figure 2 presents answers to third part of discussed study. Surveyed software developers declared, that they relay mainly on their own experience and experience of other developers. They declared really rare usage of knowledge gathered during formal education (only 15% of participants). Most of the surveyed programmers held however a masters master's degree. Only 6% of respondents never started higher education and 12% stopped studying before they graduated.

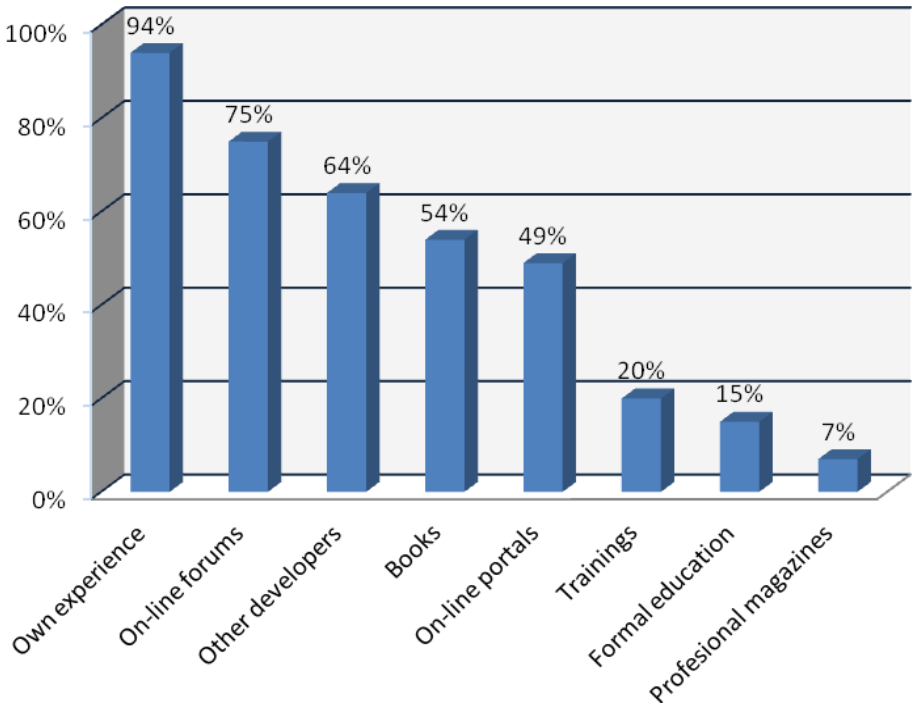

Figure 2. Sources of knowledge used at work declared by developers. (Own research)

3. CONCLUSIONS

Results of presented study are consistent with literature review. Quantitative research revealed that both soft and technical skills are important for software development process. Also some differences in perception of IT competences between academic teachers and professionals were identified. Although those differences weren't as big, as author anticipated. Only 15% of participants declared that they use knowledge gained during formal education. Based on presented results author would like to make three conclusions and suggestions:

Firstly, knowledge of algorithms is clearly overrated by academic teachers, while it has no significant influence on project outcome. Author believes that formality and mathematical background makes topic of algorithms easy to lecture, as it can be formally presented to students and relatively easy subjected to examination. Knowledge of algorithm is obviously important for programmers, however most issues related to algorithms can be explained while solving specific problems during software development.

Secondly, knowledge of versioning system is underrated by academic teachers, while it has a significant influence on development performance. In authors opinion this is also an outcome of universities tradition and boundaries. Students rarely work on large team projects. In most cases they are assigned with creating simple single-module programs. Graduates therefore have no experience with version control systems, which are necessary tools in team projects. Author believes that IT curriculums should include creation of large complex ready-to-sell projects.

Thirdly, academic teachers also underestimate managerial skills, which (as shown in Table 2) have crucial meaning for project outcomes. This could also be explained by lack of large projects in formal education. Author believes that introducing pair programming or any other form of cooperation between students would improve this competence. Pair programming could also have a good impact on abilities to share knowledge.

Question, worth answering during further studies remains: Why do so few programmers declare the use of formal knowledge, since their discrepancies in assessment of competencies do not deviate significantly from the opinions of academic teachers? Is this because teachers, despite of having good understanding of which skills should be taught, can't do it properly? Or do IT professionals fail to appreciate their formal education? Qualitative research among academic teachers could indentify limitations and factors influencing education process of young software developers, while in-depth interviews with IT professionals could lead to better understanding of their learning process.

References

[1] J. Bailey, R. B. Mitchell. *Journal of Computer Information Systems* 47 (2007) 28-33

[2] J. Cappel. *Journal of Computer Information Systems* 42 (2002) 76-82

[3] G. Coleman, R. V. O'Connor. *Journal of Enterprise Information Management* 6 (2008) 633-648

[4] R.T. Coupe, N.M. Onodu, *Journal of Information Technology* 11 (1996) 173-181

[5] J. Crawford, L. N. K. Leonard. *Industrial Management & Data Systems* 2 (2011) 164-183

[6] T. Hall, D. Jagielska, N. Baddoo. *Software Quality Journal* 15(4) (2007) 365-381

[7] T. Hung-Lian, S. Lee, S. Koh. *Journal of Computer Information Systems* 41 (2001) 76-84

[8] S. Iqbal, O. K. Harsh. *International Journal of Information and Education Technology* 2 (2013) 120-123

[9] C. K. Lee, H. J. Han. *Journal of Information Systems Education* 19 (2008) 17-27

[10] S. Lee, D. Yen, D. Havelka, S. Koh. *Journal of Computer Information Systems* 41 (2001) 21-30

[11] C.R. Litecky, K.P. Arnett, B. Prabhakar. *Journal of Computer Information Systems* 45 (2004) 69-76

[12] J. Owolabi, O. A. Adedayo, A. O. Amao-Kehinde, T. A. Olayanju. *GSTF Journal on Computing* 3 (2013) 108-111

[13] G. Sterling, T. M. Brinthaupt. *Journal of Information Systems Education* 14 (2003) 417-424

[14] R. T. Turley, J. M. Bieman. *Journal of Systems and Software*, 28(1) (1995) 19-38

[15] L. Williams, R. R. Kessler, W. Cunningham, R. Jeffries. *IEEE software* 17 (2000) 19-25

(Received 04 July 2017; accepted 20 July 2017)