PSP AND PQI: HOW DO THEY IMPROVE INDIVIDUAL SOFTWARE PROCESS

Beni Suranto

Department of Informatics, Faculty of Industrial Technology, Universitas Islam Indonesia
Jalan Kaliurang Km.14,5 Sleman, Yogyakarta 55184
Email : beni.suranto@uii.ac.id

ABSTRAK

Dalam proses pengembangan perangkat lunak, setiap pengembang baik tim maupun perorangan bertujuan untuk dapat menghasilkan produk perangkat lunak yang berkualitas tinggi. Salah satu kriteria penting dari kualitas perangkat lunak adalah jumlah kesalahan (defect) yang ditemukan pada perangkat lunak tersebut. Perangkat lunak yang berkualitas tinggi harus memiliki jumlah defect yang minimal sehingga mampu menyediakan fungsionalitas bagi pengguna dengan tingkat usabilitas yang tinggi.

Salah satu faktor penting yang berpengaruh signifikan terhadap kualitas produk perangkat lunak adalah kualitas software process yang dijalankan. Hal ini berlaku untuk proses pengembangan oleh tim maupun proses pengembangan yang dilakukan oleh software engineer perorangan. Software Engineering Institue (SEI) di Carnegie Mellon University (Amerika Serikat) telah mengembangkan metode Personal Software Process (PSP) untuk membantu para software engineer meningkatkan kualitas software process yang mereka jalankan. Selain itu, SEI juga mengembangkan Process Quality Index (PQI) yang dapat digunakan untuk mengukur kualitas software process yang dilakukan oleh para software engineer.

PSP membantu software engineer meningkatkan kualitas software process mereka melalui praktek – praktek yang mendukung proses identifikasi dan perbaikan defect pada perangkat lunak sedini mungkin. Selain itu, PSP memberikan motivasi yang besar bagi software engineer untuk dapat lebih disiplin pada setiap tahapan software process yang mereka jalankan. PQI mengukur kualitas software process dengan menggunakan indikator berupa perbandingan lama waktu penyelesaian serta jumlah defect yang ditemukan pada setiap tahapan software process.

Kata kunci: software process, PSP, defect, PQI

1. A SOFTWARE ENGINEERS IS NOT JUST A GOOD PROGRAMMER

In the IEEE Standard Glossary of Software Engineering Terminology, the term "engineering" is defined as *"the application of a systematic, disciplined, quantifiable approach to structures, machines, products, systems, or processes"* and the term "software engineering" is defined as *"the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software"*(Radatz et. al.,1990)

From the above definition, clearly that software engineering is not just about coding. And so software engineers is different from programmers. They are some certain characteristics that distinguish real software engineers from programmers.

The most important characteristic of a real software engineer is capability to produce high quality software products with minimum defects (Nguyen, 1998). Programmers who behave like real software engineers are really concerning about the quality of the software products they build for their customers. The second characteristic that makes a real software engineer is consistently improving his/her engineering performance by using defined and structured processes (Turley & Bieman, 1995). A real software engineer is a lifelong learner who never stops improving their excellence of their performance. And the next characteristic that defines a real software engineer is having the ability to make the best plan for their work based on their own experiences (Khan, 2012). A programmer can't be a real software engineer if he/she unable to learn from his/her own personal data and discover the most effective way to solve the intended problem.

2. THE PERSONAL SOFTWARE PROCESS (PSP)

Generally speaking, a software development process can be described as a process performed by software engineers to develop a software product for specific purposes. This process may be ad hoc by nature in which there are no standard guidelines or any documentations; however, it can be highly standardized as so far as high quality documentations are concerned (O'Regan, 2011). The process may be performed either by an individual software engineer or by a software project team involving many software engineers. The software process and its foundations is shown conceptually in Figure 1.

In software development, software process is very important. Most software engineers contend that the quality of a final software product adheres to that of the process so as to develop that software product (Braude & Bernstein, 2011). The better quality of the software process used in the software development, the better quality of the software product that will be produced and will be delivered to the users. It is indispensable for software engineers to enact a high-quality process in order to present or produce a high-quality product which meets their customers' needs.

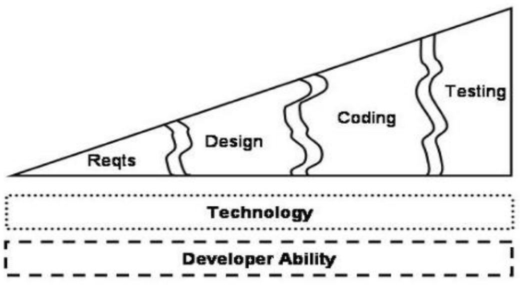

Figure 1. A conceptual View of The Software Process and Its Foundations.
(Kemerer & Paulk ,2009)

Currently, there are some well-defined software process improvement methods that can be used to improve the quality of software process. At the individual level, the Personal Software Process (PSP) is the most popular approach.

PSP which was developed in 1993 by the Software Engineering Institute (SEI) Carnegie Mellon University is a set of methods and practices that can be used by software engineers to improve their personal software process (Humphrey, 2000). There are many evidences found in many previous researches that PSP effectively improved the software engineers' personal performance. However, there are many programmers do not practicing PSP. The most common reason is that PSP requires discipline. Also, there are only small number of literature about the application of the PSP in the real software industry.

In the PSP, defect density is the most crucial indicator of the quality of the process performed by software engineers (Humphrey, 1998). It is evident that the primary goal of applying PSP is to deliver a defect free software product. To this end, software engineers using PSP remove defects in the earlier stages (i.e. design, design review, code, and code review phases) instead of removing those defects in the later stages (i.e. compile and test phases). This defect removal strategy can be enacted using the three defect filtering tactics in the PSP. Such tactics involve (1) creating a sophisticated design as coding guidance, (2) performing a disciplined review for each product in every stage, and (3) maintaining the quality of the product based on the quality process data. The structure of the PSP process is shown conceptually in Figure 2.

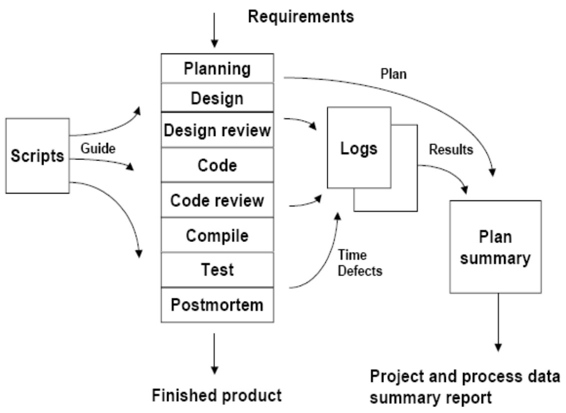

Figure 2. PSP Process Flow. (Humphrey, 2000)

The main responsibility of a software engineer is to deliver high quality products that meet the requirement from the users. Among many aspects to software product quality, the number of defects is the first thing that must be addressed (Humphrey, 1998). In fact, programmers make a lot of mistakes. In average, there was a defect found in every seven to ten LOC in a program developed by experienced programmers (Humphrey, 1998).

The PSP provides well-structured strategy to help programmers to remove almost their defects before the testing phase. For example, by performing the design review and the code review steps in the PSP2 programmers can find and correct the defects earlier. In general, for systems which were built with the PSP method there are only 0.2 defects per KLOC found in the testing phase (Humprey, 2005). In the paper entitled "Result of Applying the Personal Software Process", the authors showed the value of the PSP which was used in three industrial software groups (i.e. Motorola Paging Products Group, Advanced Information Services Inc., and Union Switch & Signal Inc.). They found that the PSP helped the three industrial software groups to efficiently produce the better software by improving their planning and scheduling activities and reducing the development time (Ferguson, 1997). In addition, Prechelt et al. (2000) found that compared with the non-PSP-trained programmers, the PSP-trained programmers produced programs which were much reliable for the same tasks.

As it was mentioned before, software engineering or software development is not such a simple process. It consists of many activities and involves many potential risks (e.g. time, money). A poor software process will lead to a failure. So, improving the software process is an important thing for programmers. Moreover, most programmers work in a team and each of them has contributions to the team. That means the better individual software process performed by each programmer, the better performance of the team.

Every engineer responsible for his/her own engineering process. In order to improve their engineering process, real engineers use well-defined and measured methods with appropriate tools (Khan, 2012). This agrees with the logic for the PSP that defined and structured processes will result in efficiency. Also, the PSP was developed with the idea that programmers should learn from their own experiences to consistently improve their software process (Humprey, 2000).

The PSP has four maturity levels (i.e. PSP0, PSP1, PSP2, and PSP3) represent four different process levels to guide programmers in improving their individual software process The PSP life cycle phases is shown in Figure 3.

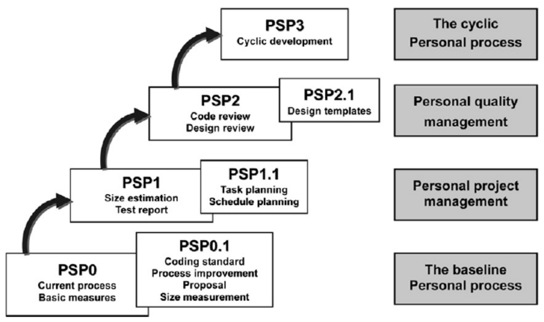

Figure 3. PSP Life Cycle Phases.
(Shen et. al., 2011)

It is true that practicing the PSP requires discipline. However, one of the most important factors for software quality is process discipline. Shen et. al. (2011) found that the performance of programmers who used disciplined software processes was superior than that of other programmers who used ad hoc processes.

Since it is very often that software development must be accomplished within a restricted budget and time, planning and estimating skills are very important for programmers. The PSP has been proven to increase the accuracy of the programmers' estimation on the time they need for developing a program for a particular task (Prechelt, 2001). It means that by conscientiously using the methods and the practices provided by the PSP, programmers will be able to make the best plan for their works based on their personal data from their previous experiences.

3. THE PROCESS QUALITY INDEX (PQI)

Many researchers in the software engineering field have undertaken many studies and experiments and have proposed some methods and/or tools for improving a software development process. Software Process Improvement (SPI) is perhaps the most powerful method for improving the software development process both at individual level and at organizational level. SPI takes the advantages of some well-established process models (e.g. CMMI, PSP, TSP, Six Sigma, and ISO 9001 Standard) to be adopted by software engineers to improve their individual or organizational performance so they can achieve their primary goal; that is, to produce a high quality software product (O'Regan, 2011).

When software engineers apply a particular process model for their software development process, it is necessary for them to perform an evaluative mechanism to know whether we achieve a high quality software development process or not. To do such an evaluation, at the outset, they need to know what a high-quality software development process looks like. In the other words, they need to define the criteria of a quality software development process. The Process Quality Index (PQI) can be used as a metric to define the quality of their software development process. PQI which was proposed by the SEI has been a metric or a yardstick for evaluating the quality of a particular software development process. PQI was derived based on the characteristics of process in the PSP (Humphrey, 2005).

PQI determines the process quality by considering three things that software engineers can measure at a particular software development process stage. These things embrace the size of code they produced, the time they spent, and the number of defects they removed from their code at that stage (Humphrey, 2005). PQI provides a quality profile that can serve as a benchmark for the quality of the entire software development process. Fundamentally, there are three software engineering principles embodied in the PQI's quality profile as listed below:

a. Design is of paramount importance in the process.
b. Technical reviews serve as the foundation of quality.
c. The number of test defects is predictive of that of defects in the delivered product.

To evaluate the quality of a software development process based on the PQI's quality profile, software engineers use the following items (Humphrey, 2005):

a. Design/Code Time = Minimum (design time/coding time: 1.0).
b. Design Review Time = Minimum (2 * design review time/design time: 1.0).
c. Code Review Time = Minimum (2 * code review time/coding time: 1.0).
d. Compile Defects/KLOC = Minimum (20/(10 + compile defects/KLOC):1.0).
e. Test Defects/KLOC = Minimum (10/(5 + unit test defects/KLOC):1.0).

The combination of the five items above will result in a metric score between 0.0 and 1.0 which represents the level of the quality of the software development process. The higher the PQI metric score of a software development process is, the higher quality of that process will be.

PQI can be considered as a good choice when evaluating the quality of a software development process since it effectively captures all important aspects of the process. Based on the five dimensions of the PQI, it is evident that PQI covers both the creative phases (i.e. design and coding) and the review phases (i.e. design review and code review). Furthermore, it evaluates the most important quality criteria of a software product called the defect density.

PQI provides an efficient way to evaluate a software development process. The five criteria used in the PQI metric can be computed when each stage is completed. PQI is the most appropriate metric for evaluating the quality of a software development process in as much as it makes use of the three realistic and relevant measurements as defined in the PSP (i.e. time, size, and number of defects).

Whether a PQI metric of 1.0 represents a high quality software development process deserves further discussions. In these discussions, I would like to discuss this issue based on the five dimensions of the PQI's quality profile.

3.1. An Adequate Design Time Means High Quality Design

Design plays a pivotal role in every engineering process, including in software engineering (producing high quality software), civil engineering (building a bridge), aircraft engineering (assembling an airplane), electrical engineering (developing an electrical power plant), and mechanical engineering (building a supercar). In a software development process, software engineers will be unable to produce a software product that meets the intended requirements without a proper design.

Thus, it is obvious that design is a compulsory stage in a software development process, but it is important to ponder whether software engineers perform a good design task. In response to this challenge, they can deploy PQI as a metric for ensuring the quality of their design process using the ratio of the design time and the coding time. This approach does make sense in that the only measurement that they can do in the design phase is the time spent on it. The next question to ponder is how much the time they need to spend to be able to perform a high quality design process. Adhering to PQI, a good design process must take more than half time of the corresponding coding process.

3.2. An Adequate Design ReviewTime Means High Quality Design

As suggested by the principle of the PSP, software engineers need to remove defects at the earlier stages of their software development process. At the first stage, they utilize design review to identify and remove defects of the design. They evaluate their design using a predefined design review checklist. To be able to effectively evaluate the design, they are required to have a design review checklist that touch upon all important aspects of software design to guarantee that the design is complete (i.e. the design meets all relevant requirements), consistent (i.e. there are no contradictions in the design), correct (i.e. the design shows that the product will perform the intended function and uses the correct logics), robust (i.e. the design addresses all fault-related requirements), understandable (i.e. the design has no ambiguity), and verifiable or testable (i.e. the design can be verified and/or tested) (Nelson & Schumann, 2004).

Table 1 shows the PSP data from 3.240 program tasks completed by experienced software engineers. These data was analyzed by the SEI for defining the characteristics of a high-quality software development process (Humphrey, 2005).

Table 1. *PSP Data Analyzed by The SEI.*

Phase	Hours	Defects Injected	Defects Removed	Defects/ Hour
Design	4,623.6	9,302		2.0
DLDR	1,452.7		4,824	3.3
Code	4,179.6	19,296		4.6
Code Review	1,780.4		10,758	6.0

In the design review phase, software engineers can measure two things: the time they spent on it and the number of defects they found during the phase. As seen in Table 1, during the design phase, professional software engineers inject about 2.0 defects per hour while in the design review phase they find about 3.3 defects per hour. Mathematically speaking, it could be calculated that the software engineers spent roughly 36 minutes (60 * 2.0/3.3) to review the design in order to find the defects in the design phase. From this calculation, the ideal time for design review is at least half of design time. Thus, if a PQI metric obtains 1.0, a high quality design review process is properly performed.

3.3. An Adequate Code Review Time Means High Quality Code

Once software engineers finish their code, they need to perform design review to evaluate the code and remove defects in their code. Bacchelli & Bird (2013) found that almost all the software engineers included finding defects as one of the reasons for doing code reviews.This finding is shown in Figure 3.

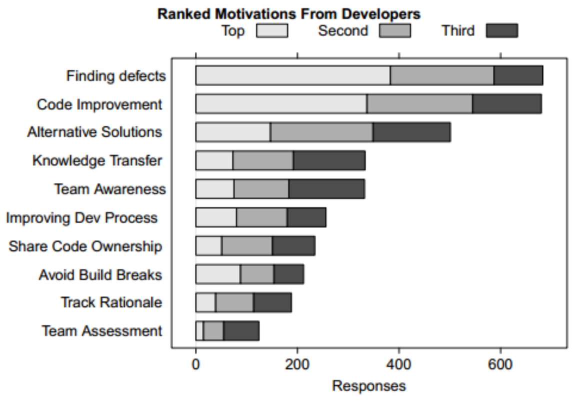

Figure 3. Software Engineers' Motivations for Code Review. (Bacchelli & Bird, 2013)

In PSP, software engineers use programming-language specific view for the design review. In doing so, they must ensure that their code is correctly enacted based on the programming language they use in the entire software development process. This code review requires the following standard checklist (Nelson & Schumann, 2004) :

a. Generic properties: strategic and tactical comments in the code. Strategic comments belong to a function or procedure while tactical comments explain a particular single line of code.
b. Variables and data types properties: variable declarations and initializations, constants, variables naming.
c. Object-oriented related properties: classes, method overriding, inheritance, class interface.
d. Flow control: iteration structure, method calls, decision control structure, recursive structure.
e. Computation: values calculation, variables update.
f. Error handling: exception type, error messages.
g. Argument passing: methods' argument declaration, return values, temporary objects.
h. Coding standards: indentations, code block structure format.

These properties of code take considerable time for software engineers to review the code. Based on the PQI metric, they need to spend more than half of coding time for the code review. I will elaborate on a conceptual justification for this.

From the data shown in Table 1, on average, a professional software engineer will inject approximately 4.6 defects in one hour of coding. Table 1 shows that they can find roughly 6.0 defects in one hour during the code review. In other words, the time taken to review the code produced in one hour coding is about 46 minutes ((4.6/6.0) x 60). From this calculation, the ideal time for code review is at least half of coding time. This suggest that if software engineers obtain a PQI metric of 1.0, they perform a high quality code review process.

Kemerer & Paulk (2009) investigates the effect of review rate on defect removal effectiveness and the quality of software products, while controlling for a number of potential confounding factors. Two data sets of 371 and 246 programs, respectively, from a PSP approach were analyzed using both regression and mixed models. Review activities in the PSP process are those steps performed by the developer in a traditional inspection process. The results show that the PSP review rate is a significant factor affecting defect removal effectiveness, even after accounting for developer ability and other significant process variables.

3.4. Minimum Defects In The Final Stage Means High Quality Product

There is a myriad of different definitions of a high quality software product. In a technical sense, a high quality software product is the one that contains a minimum number of defects in the final stage of its development process (Gillies, 2011). In this process, there are two determinants of assessing if the final product is of high quality. These two determinants include compile defects and test defects.

The PQI uses the number of compile defects and test defects to represent that of defects at the final stages of a software development process. A compile defect is the one removed during the code compilation phase while a test defect is that removed during the testing phase.

The SEI indicates that when a software product contains more than about 10 compile defects per KLOC in the compiling phase, it has a mediocre quality in the testing phase. The SEI also shows that when a software product contains more than about 5 test defects per KLOC in the unit testing phase, it has a poor quality in the system testing (Humphrey, 2005). In addition, from data on many software products, the SEI reveals that when a software product obtains less than 10 compile defects per KLOC and less than 5.0 test defects per KLOC, it has a very few if any remaining defects (Humphrey, 2000). These findings support the fourth and the fifth criteria of the PQI metric. Such findings show evidence that when a software engineer achieves a PQI metric of 1.0, he/she yields a high quality software product.

4. CONCLUCIONS

Programmers who conscientiously apply the PSP in all their projects are behaving like real software engineers. There are some certain characteristics that distinguish real engineers from programmers. Programmers will be able to gain the real software engineer characteristics by conscientiously using the PSP in their software engineering process. In this context, having a real engineers' characteristics means increasing the quality of the product, improving the individual performance, and having the ability to make the best plan based on the previous data.

PQI provides a valid and trustworthy yardstick for evaluating the quality of a software development process. The PQI metric take into account all essential aspects in the software development process, and it adheres to a relevant and doable measurement mechanism to determine the criteria of the quality software development process. The highest score in the PQI metric of the software development process is 1.0. I have shown four arguments to support the validity of the scoring calculation employed in the PQI metric. Each of the arguments shows how each of the criteria in PQI metric properly represents the quality of each of the defect filtering stage deployed in the whole software development process. This representation is conceptually grounded in the PSP framework.

REFERENCES

Bacchelli, A., & Bird, C, Expectations, outcomes, and challenges of modern code review. In *Proceedings of the 2013 International Conference on Software Engineering* (pp. 712-721). IEEE Press, 2013.

Braude, E. J., & Bernstein, M. E.. *Software engineering: modern approaches*. J. Wiley & Sons, 2011.

D'Ambros, M., Bacchelli, A., & Lanza, M., On the impact of design flaws on software defects. In *Quality Software (QSIC), 2010 10th International Conference on* (pp. 23-31). IEEE, 2010.

Ferguson, P. "Results of applying the Personal Software Process". Computer (Long Beach, Calif.) (0018-9162), 30 (5), p. 24. DOI: 10.1109/2.589907, 1997.

Gillies, A. *Software quality: theory and management*. Lulu. Com, 2011.

Humphrey, W. S. The software quality profile. *Software Quality Professional*, *1*(1), 8-18. Diakses dari *http://citeseerx.ist.psu.edu/-viewdoc/download?doi=10.1.1.174. 1021&rep=rep1& type=pdf*, 1998.

Humphrey, W. *The Personal Software Process^{SM} (PSP^{SM})(CMU/SEI-2000-TR-022, ADA387268)*. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University. Diakses dari *http://www.sei.cmu.edu/reports-/00tr022.pdf*, 2000.

Humphrey, W. The Personal Software Process^{SM} (PSP^{SM})(CMU/SEI-2000-TR-022, ADA387268). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, Diakses dari *http://www.sei.cmu.-edu/reports/00tr022.pdf*, 2000.

Humphrey, W. S. "Acquiring Quality Software." *CROSSTALK The Journal of Defense Software Engineering* 19-23, 2005.

Humphrey, W. S. *PSP^{sm}: a self-improvement process for software engineers*. Addison-Wesley Professional, 2005.

Kemerer, C. F., & Paulk, M. C. The impact of design and code reviews on software quality: An empirical study based on PSP data. *Software Engineering, IEEE Transactions on*, *35*(4), 534-550, 2009.

Khan, AK "Amalgamation of Personal Software Process in Software Development Practice". *And lo, the star*, 1 (2), p. 59. Diakses dari *http://www.starjournal.org/uploads/ starjournal/07.pdf*, 2012.

Nelson, S., & Schumann, J. What makes a code review trustworthy?. In *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on* (pp. 10-pp). IEEE, 2004.

Nguyen, D. Q. The essential skills and attributes of an engineer: a comparative study of academics, industry personnel and engineering students. *Global J. of Engng. Educ*, *2*(1), 65-75. Diakses dari *http://citeseerx.ist.psu.edu/viewdoc/ download?doi=10.1.1.124.1502&re p=rep1&t ype=pdf*, 1998.

O’Regan, G, Motivation for Software Process Improvement. In *Introduction to Software Process Improvement* (pp. 1-12). Springer London, 2011.

Paulk, M. C. Factors affecting personal software quality. Institute for Software Research, 4. Diakses dari *http://repository.cmu.edu/cgi/-viewcontent.cgi?article=1011&context=isr&sei-redir=1&referer=http-%3A%2F%2Fscholar.google.com.au%2Fscholar%3Fstart%3D10%26q%3DPSP%2Band%2Bprogrammers%2Bbehavior%26hl%3Den%26as_sdt%3D0%2C5%26as_ylo%3D1995#search=%22PSP%20programmers%20behavior%22*, 2006.

Pomeroy-Huff, M., Cannon, R., Chick, T. A., Mullaney, J., & Nichols, W. *The Personal Software ProcessSM (PSPSM) Body of Knowledge, Version 2.0* (No. CMU/SEI-2009-SR-018). CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, 2009.

Prechelt, L. "An experiment measuring the effects of personal software process (PSP) training". IEEE transactions on software engineering (0098-5589), 27 (5), p. 465. DOI: 10.1109/32.922716, 2001.

Radatz, J., Geraci, A., &Katki, F. IEEE standard glossary of software engineering terminology. *IEEE Std, 610121990*, 121990. Diakses dari *http://www.idi.ntnu.no/grupper/su/publ/ese/ieee-se-glossary-610.12-1990.pdf*, 1990.

Shen, W. H., Hsueh, N. L., & Lee, W. M. Assessing PSP effect in training disciplined software development: A Plan–Track–Review model.*Information and Software Technology*, *53*(2), 137-148, 2011.

Turley, R. T., &Bieman, J. M. Competencies of exceptional and nonexceptional software engineers. *Journal of Systems and Software*, *28*(1), 19-38. Diakses dari *http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.54.8463&rep=rep1&ty pe=pdf*, 1995.