**DOES SWEBOK REALLY HELP TO FIGURE OUT**

**ACTIVITIES OR TASKS OF KNOWLEDGE AREAS OF SOFTWARE ENGINEERING?**

Software engineering plays an important role for overall development of new hires in corporate training and in our lives when we are interacting with the IoT devices which has definitely software engineering background. According to Kharagpur I., “Software engineering course plays an important role for overall development of new hires in corporate training. During last few years, both academia and industry turned their attention towards integrating soft skills in learning, while efforts are being made to bridge the knowledge, skill gap between industry requirements and what academic institutes churn out” (Boyeena, M. and Goteti, P., 2011). To fill the gap between industry requirements, IEEE developed and evolved the SWEBOK standards to provide a consensually validated characterization of the bounds of the software engineering discipline and to provide topical access to the Body of Knowledge supporting those disciplines to be useful in global. This essay will advance the claim that SWEBOK (or some another framework that contains knowledge areas) helps to figure out what should be known, and how to perform activities or tasks of the knowledge areas by arguing both the ideas aligned with it and not aligned with it.

One of the strong arguments about the skills in software engineering is that the guidance of SWEBOK helps figuring out the subset of generally accepted software engineering skills knowledge for software engineers coming from different backgrounds and levels of experience learned common tasks and activities in a standard way to create a good collaboration in the projects around the World. Alongside global connectivity, the software development process is a complex process requiring the effective collaboration of many software engineers (Ludi S. & James C., 2001). The teaching styles, curriculums, activities and tasks that are being taught to the computer science department students are not in the same format or not in collaboration with the overall World universities. Concerning this, when undergraduates that are coming from different backgrounds that have enriched with the different approaches of solving problem in the constraints of the academic environment and when the graduates that have different job experience in a typical industrial Project want to work together, they can talk in the same language in terms of figuring out activities and tasks that are placed in the topics under the 15 knowledge areas of SWEBOK. This inevitable reliance on software in current society and the need for software development and maintenance at a massive scale have forced the industry to employ people from widely varied backgrounds and skills during the entire life cycle of software development. Emerging disciplines like end-user-software-engineering encourage end-users to develop and author software for their needs. This demanded to SEE community to induce software engineering skills to undergrads, graduates, professionals and end-users such that they can deliver quality software (Chimalakonda, S. & Nori, V., 2011). Therefore, SWEBOK is intended as a guide to the subset of generally accepted software engineering knowledge to form common principles that enable engineers to work in the same academic or industrial Project together easily and within an effective collaboration.

Furthermore, as we dive into the standardization of generally accepted knowledge in software engineering by SWEBOK as it previously mentioned, also there is another strong argument about skills in software-engineering is that SWEBOK organization also helps software engineers to avoid errors or to prevent encountering semantic glitches in the software project life cycle. It was also stated in the original SWEBOK that the benefits of software engineering standards in SWEBOK are many and include improving software quality helping avoid errors, protecting both software producers and users, increasing professional discipline, and helping the technology transition. At the business level, software properly applied to a problem can eliminate months of work and translate to elevated profits or more effective organizations. Moreover, organizations that acquire or provide successful software may be a boon to the society in which they operate by providing both employment and improved services. At the societal level, direct impacts of software success or failure include or exclude accidents, interruptions, and loss of service. Indirect impacts include the success or failure of the organization that acquired or produced the software, increased or decreased societal productivity, harmonious or disruptive social order, and even the saving or loss of property and life ( Bourque, P., & Fairley, E. R., v. 3.0, 2014, p.11:4 & p.11:5). Thus, Software engineering standards establish guidelines for generally accepted practices and minimum requirements for products and services provided by a software engineer, and all of those provided with the minimum chance of errors, a healthy software project life cycle, expected results both at the business and societal level.

On the other hand, despite the generally positive perceptions about SWEBOK, there are some opposing claims that show SWEBOK has several missing points or should be updated to compete with other universities' regularly updating domain and curriculum. According to Bloom’s Taxonomy, they looked at the differences found between taxonomies analyzed. The complete analysis can be found in Dolog et al (2016, p. 19:9). They realized that in some areas of the software engineering curriculum the students at Aalborg University achieve a higher level of competencies than those expected for the curriculum in the SWEBOK (GSwE2009, respectively). By looking at the application domain and knowledge from those areas, they found indication that in the following areas of software requirements fundamentals, students at Aalborg University achieve the analytical level of Bloom’s taxonomy instead of the comprehension level: Deﬁnition of software requirements, product and process requirements, functional and nonfunctional requirements, emergent properties. As a result of this Bloom’s taxonomy research they expressed that the Graduate Software Engineering 2009 Curriculum Guidelines for Graduate Degree Programs in Software Engineering (GSwE2009) [Pyster 2009] introduce a few categories not in the SWEBOK, such as social, legal, and historical issues, including data conﬁdentiality and security, surveillance, and privacy. In this category, students at Aalborg University score lower at the knowledge level in comparison to the recommended comprehension level, partly because these topics are not in the curriculum (Dolog, P., Thomsen, L.L., Thomsen, B, 2016).Therefore from the skills in software-engineering perspective SWEBOK should be updated as the new and better curriculums discovered in computer society.

In conclusion, even if the SWEBOK has some missing points or again some points that need to be improved and updated, it can be seen obviously that it helps many software engineers to figure out what should be known, and how to perform activities or tasks of the knowledge areas by providing them global reliance on software society and a healthy software project life-cycle without errors and problems. In addition, we can continue to spend much more time to understand how we can use SWEBOK to make it more useful about understanding skills in software engineering, and to complete the missing parts so that we can get much more powerful tool to compete with other curriculums that are aiming to teach those skills.

**References**

1. Alarifi, A., Zarour, M., Alomar, N., Alshaikh, Z., Alsaleh, M. (2016), SECDEP: Software Engineering curricula development and evaluation process using SWEBOK.*Elsevier B.V.*
2. Boyeena, M. and Goteti, P. (2011), A Blended Approach to Course Design and Pedagogy to Impart Soft Skills: An IT Company’s Experiences From Software Engineering Course. *Proceeding of the 2011 IEEE Students' Technology Symposium.*
3. Chimalakonda, A. and Nori V. (2011), Can We Make Software Engineering Education Better By Applying Learning Theories? *International Institute of Information Technology – Hyderabad, Hyderabad.*
4. Peter Dolog, Lone Leth Thomsen, and Bent Thomsen. 2016. Assessing problem-based learning in a software engineering curriculum using Bloom’s taxonomy and the IEEE software engineering body of knowledge. *Aalborg University.* *ACM Trans. Comput. Educ. 16, 3, Article 9 (May 2016), 41 pages.*
5. Bourque, P., & Fairley, E. R. (2014). SWEBOK. (v. 3.0,p.11:4 & p.11:5 ).*IEEE Publications.*