Brief Report

An Experience Report on Designing and Delivering of a Software Engineering Education Module to Enhance Employability Skills

SOUMYA Sankar BASU *

Posted Date: 22 October 2024

doi: 10.20944/preprints202410.1690.v1

Keywords: software engineering education; employability; module design

Article

An Experience Report on Designing and Delivering of a Software Engineering Education Module to Enhance Employability Skills

Soumya Sankar Basu [1,*]

[1] School of Computing and Digital Technologies, Sheffield Hallam University
* Correspondence: soumya.basu@shu.ac.uk

Abstract: Software Engineering education has grown during the past couple of decades as a specialized branch of computer science education to address employment demands. Software Engineering education is constantly evolving because of its dynamic nature. However, there are still some gaps between the skills being taught in the universities and skills required for employment. In this paper, we present our experience of designing and delivering a new module to enhance skills for employment. We have taken a structured approach to analyze the gap through literature review and our personal experiences of engaging with employers of our students. Based on the analysis, we arrived at a set of gaps between what we teach and what the employers of our students need. We prioritized the gaps and envisaged a new module 'Contemporary Software Engineering' to address the gaps and subsequently help our students for an easy transition to job. As part of our experience, we present the background, design and delivery of the module and reflect on our journey of delivering the module for a few years.

Keywords: software engineering education; employability; module design

1. Introduction

Software engineering education programs have evolved from computer science programs in last few years to cater employment demands. Software engineering programs continue to focus on theoretical and technical computer science topics [1]. This seems to create a gap between software engineering skills learnt in the university and the skills needed in employment [1].

Different research highlights the gaps from different perspectives. In 2000 Shaw outlined a software engineering education roadmap [2]. The roadmap focused on moving away from the content and considering how the curriculum could focus on different software engineering roles.

An investigation on graduate software developer experience in workplace reported that the students generally demonstrate good design skill, but lack other important skills related to software engineering [3]. This investigation has been carried out in one large organization.

There are many investigations highlighting the gaps. A later investigation [4] presented that student lacked software engineering skills like testing, configuration management, use of proper tools, problem solving, requirement, development process, communication etc. from software engineering point of view. Another investigation [5] concluded that graduating students do not possess skills to deliver large scale software engineering projects. This is very much aligned with the earlier investigation.

The investigation presented in [1] considers data from 12 countries across US, Canada, Europe, Asia, Africa, and Australia and presents a global view of the gap between academic skills acquired through software engineering education and skill requirement in employment. The most important gaps are identified in the areas of Software Engineering models and methods, Software Engineering process, Design and Architecture, Requirement, Quality, Configuration management and testing.

 doi:10.20944/preprints202410.1690.v1

Other investigations also hint at similar types of gaps between software engineering education and requirement of employers. The presented investigations consider data across the globe. Consequently, we can summarize the gaps as a baseline of gaps between education and employment.

The seven base line gaps are a) importance of role playing in the curriculum, b) software engineering method & development process, c) requirement & problem solving, d) design & architecture, e) configuration management & testing, f) quality and g) communication.

United Kingdom is no exception in terms of gaps and the same is confirmed by [6]. It presents high unemployment of computer science graduates versus the high demand for such skills. Most of the topics covered in software engineering courses in UK universities concentrate on different aspect of programming, programming paradigms, designs, and mathematical foundation. A study on what is taught while teaching software engineering in UK universities [7] also concluded that a) software engineering is not all about programming and b) software engineering graduates fill in different roles, each of which need specific content.

The studies mentioned above, and similar studies consider industry as one bucket. From practicing experience as a software engineer for more than two decades, the author appreciates the difference between different type of industries e.g., software development organization, consulting organization, start-up organization, large organizations, Small and Medium Enterprises etc. Keeping the difference in mind, we conducted an analysis on the kind of companies our graduates are engaged with as employees or placement students. From the analysis, we prioritized the gaps that we need to address to facilitate our students' better enablement for employment. We envisaged the 'Contemporary Software Engineering' module to address those gaps

The following section describes the background study to prioritise the gaps. The subsequent two sections describe the module design to address the gap and the module delivery mechanism. The discussion section reflects on the lessons learnt from first few occurrences of the module and how students are perceiving this module. The paper concludes with a conclusion section summarising the experience and outlining future enhancements.

2. Background Study

To prioritize the gaps, we analyzed the kind of organizations our students engage with as employees or placement student. Our baseline analysis is based on internally published data for 2018-19 by our university employability department.

The employment data for software engineering and computer science graduates for 2018-19 are depicted in the following chart. It is based on the number of students responding to the employability questionnaire.

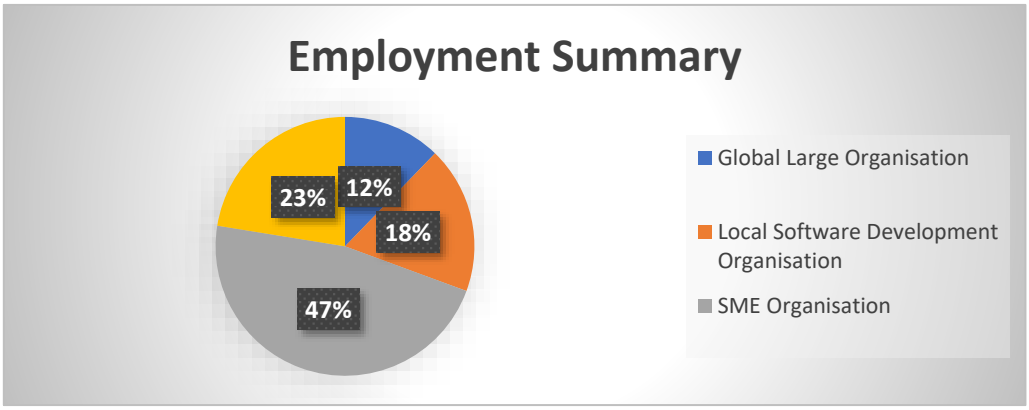

Figure 1. Employment Summary.

The placement data for software engineering and computer science graduates for 2018-19 are depicted in the following chart

 doi:10.20944/preprints202410.1690.v1

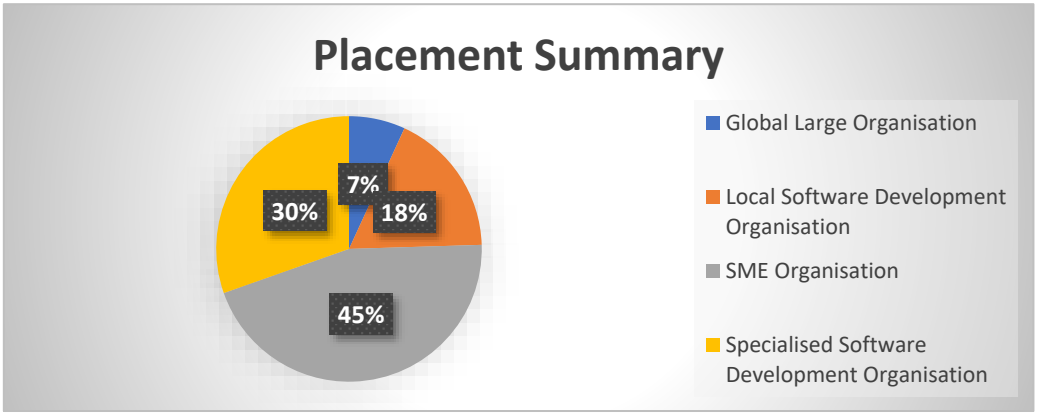

Figure 2. Placement Summary.

Both placement and employment data shows that most of our students engage with SME organizations, Local software development organizations and specialized software development organizations. SME organization shares the largest percentage of students. Very few students get engaged with large global organizations.

By SME organizations, we mean small and medium enterprises specializing in different business and not having any software development experience. Local software development organizations are those who develop software for different clients within the UK and based locally inside the UK. We segregated specialized software development organizations from local software development organizations based on the nature of software development. Specialized software development organizations develop software targeted to a specific industry such as education or healthcare.

As part of the background study, we tried to identify the challenges faced by SME organizations through literature review and our engagement experience with different local SME organizations to prioritize the seven base line gaps presented in the introduction.

Though we have presented local software development organizations and specialized software development organizations as separate categories, both are software development organizations on different scales. We assume their challenges are captured in investigations like large global organizations. From our experience of engaging with local SME organizations, we have found out that most of them are like start up organizations from a software development perspective. So, to understand their challenges we relied on literature review and our experience.

Melegati et al identified major challenges of software start-ups and suggested some way forwards to overcome those in their chapter [8]. They have highlighted on 'thriving in technology uncertainty' and 'time spent developing features that users are not interested in' as two major challenges. The chapter has also emphasized team related challenges like 'lot of activities in a short time' and building entrepreneurial team'. The chapter recommends getting help of methodology from Lean Startup [9] to avoid mistakes already done by other people.

'Thriving in technology uncertainty' can be mapped to 'design and architecture' from the baseline gaps as 'design & architecture' focuses on technical solution. 'Time spent developing features that users are not interested in' can be mapped to 'requirement & problem solving'. Team related challenges can be mapped to 'software engineering method & development process'.

Melegati, J et al [10] analyzed four start-up organizations and discussed the inhibitors and enablers. Finally, they concluded that experiment driven software development as an important approach can help software start-up succeed. Clearly experiment cannot be done unless the requirement is not clear, and as a result it can be mapped to 'requirement & problem solving'.

Gupta V et al [11] studied software start-ups from US, UK and India and identified three major challenges a) handling software evolution, b) releasing product to the market and c) software engineering. There are other similar studies like [12] hinting at the same factors.

 doi:10.20944/preprints202410.1690.v1

'Handling software evolution' maps to 'design & architecture' from our identified baseline in the introduction. 'Releasing product to the market' maps to 'requirement & problem solving'. 'Software engineering' directly links to 'software engineering method & development process'.

From literature review, we can summarize the challenges faced by start-up organizations as 'design & architecture', 'requirement & problem solving' and 'software engineering methos & development process'. Now we will reflect on our engagement experience with local SME organizations to understand their challenges. We from our university engage with local SME organizations through different knowledge exchange programs Knowledge Transfer Partnership [13]. Let us present some engagement examples

As part of Knowledge Transfer Partnership, we worked with a document management organization. They had a home-grown project management software developed using legacy technologies and they wanted to build a digital platform to handle all business operations for their organization. Before this project they did not have any in-house software development team. We had to establish requirement management techniques, define software development methods and create an architecture to develop the solution satisfying the requirements.

As part of another program, we engaged with another SME organization implementing six sigma to optimize process for manufacturing clients. They were aware that they needed software to help them scaling up but were not aware of what the software should do. We helped them formulating in understanding requirement that can solve their problem using design thinking approach.

To site another example, we engaged with a mechanical engineering organization to review a proof-of-concept IoT solution that they built to monitor health parameters in a mechanical production process. The organization wanted to review the solution before building it on a large scale. We found that the solution works for the purpose, but the architecture was not robust (e.g., they were using two different cloud service providers for doing different things without any rationale). Their proof of concept was developed by one developer and there was no software engineering process to scale up the development team.

We have similar other examples. Our engagement experience identifies the challenges faced by local SME organizations as 'design & architecture', 'requirement & problem solving' and 'software engineering methos & development process'. This is aligned with our findings from literature review about the challenges of start-up organizations.

We prioritized addressing 'design & architecture', 'requirement & problem solving' and 'software engineering methos & development process' to be addressed in our module. However, design has been taught in other programming modules. So, we addressed the architecture more. We did not prioritize 'configuration management & testing' as that is covered in other project modules. Though 'quality' is not part of the challenges faced by SME organizations, we prioritized that for those students who get engaged with large organizations. We decided to address 'role playing' and 'communication' as part of the module delivery as opposed to design.

The next section describes the content design of the module demonstrating how we addressed the prioritized four gaps. The module delivery section highlights how we address the other two prioritized gaps

3. Module Design

As per our prioritization we designed the content of the module 'Contemporary Software Engineering' keeping 'requirement & problem solving', 'architecture', 'software engineering methos & development process' and 'quality' in mind.

Initially we offered this module to final year software engineering students. We chose the final year, as then students will have an idea about programming, design and development. Additionally, some of our final year students go for one year placement before concluding final year. We envisaged that they would be able to appreciate the module and can share their placement experiences with others to make the module more effective.

 doi:10.20944/preprints202410.1690.v1

As part of the module design, we referred to the modules taught in previous years to avoid potential overlap. We provided a few problems for the students to work with. The problems are chosen in such a way that students can relate to them easily as a user. For example, keeping international students in mind two of the problems given were a) Finding the right course in the right university for a student and b) Immigration helper for international students. Students were expected to choose one of the problems and identify requirements, arrive at a technical solution and design a software engineering method to develop the solution for the problem. Students would be allowed to suggest their own problem as well.

Before final year, students learn documenting a given set of requirements. They do not get much opportunity to arrive at a requirement from a problem description and arrive at a solution for the problem. So as part of this module, we introduced a widely used contemporary technique of design thinking [14] to identify requirements empathizing with users and find the right solution to the problem. Design thinking is one of the most recognized approaches that drives innovation and creates a solution that end users need [15].

We decided to leverage IBM enterprise design thinking approach [16] for practicing design thinking in the class, as this approach has been harvested from various client engagements. The requirement & problem solving is handled through the introduction of design thinking.

After requirement identification, the module introduces how to arrive at an initial solution architecture. The module introduced reference architecture [17]. The module introduced reference architecture for different technologies and domains and provided guidance on how to choose a reference architecture and leverage that to arrive at a software architecture to solve a particular problem.

We planned to introduce software engineering method, after architecture. There are different software development lifecycles, but the lifecycles are not prescriptive. They need tailoring to arrive at a software engineering method for developing a solution, based on project characteristics. The module aimed to enable students about tailoring to arrive at the method.

R Akbar [18] identified that developers and project managers do not have proper guidelines to tailor a method. He tried to suggest a guideline for tailoring an agile process considering software project characteristics as an important factor. There are similar guidelines available, but as the module needs to be lifecycle agnostic it did not consider those. The author has more than two decades of experience tailoring and arriving it software development method for different engagements. His practical experience is leveraged in designing the module content.

We highlighted a contemporary practice driven approach of software engineering, where students can learn about different practices, the tools that help realizing the practices, and how to decide about the applicability of the practices in a particular situation. The module leveraged IBM Garage method [19] as a reference for practice driven approach, as a) the method is harvested from many project experiences and gets continuously updated from applied experience and b) documentation is available in public domain. The module also introduced Lean start up methodology [20] and discussed how it can be applied to optimize software engineering methods for developing a solution. This part of the module aims to enhance students' practical understanding of Software Engineering Methods & Development Process.

Finally, to facilitate students' understanding of software quality, the module introduced ISO 9000 family [21]. As the module is specifically for software engineering, more stress was given on the key process areas of Software Engineering Institute Capability Maturity Model [22].

Initially the module was assessed using two parts, 70% coursework and 30% examination. Later based on delivery experience and student feedback, we changed the assessment as 100% coursework.

The next section describes the delivery modality of the module highlighting how we address 'role playing' and 'communication'.

3. Module Delivery

To encourage 'communication' and 'role playing', we follow a workshop-oriented approach to deliver this module as opposed to traditional lecture & lab-based approach.

 doi:10.20944/preprints202410.1690.v1

At the beginning, we request students to form a group of 3 to 5 members. The group works as a start-up organization and chooses a problem from the given problems. They are allowed to suggest their own problem as well. Each company was supposed to run design thinking to arrive at a requirement and innovative solution for their chosen problem, arrive at initial architecture and finally decide on the software engineering method to develop the solution for the chosen problem.

We plan the classroom as an informal environment with round tables with 4-6 students at one table, no fixed computer with flexibility of using laptops, flipcharts, post-its, and several whiteboards. This informal environment is used for simulating an environment to help in conducting effective workshops.

Members of each group get the opportunity to play different roles like project manager, business analyst, software architect etc. Group members are free to rotate their roles throughout the semester. During design thinking workshop, group members also play roles of different personas. This encouraged role playing.

Each workshop runs for two hours. First 30 to 45 minutes are used to discuss the topic for the workshop. The discussion starts with a small recapitulation of the last workshop's topic. After the discussion about the topic, each group works on the topic within their group for about 30 to 45 minutes. The workshop ends with a playback session for the whole class, where each group presents their work in an informal manner to the whole class and other groups provide feedback. This encourages students communicate to within and outside their group.

The workshop model facilitates a collaborative learning environment. We encourage students to finish their coursework within the scheduled timings. We also allow students to submit pictures of informal outputs like flip chart/ post-it etc. The workshop plan has built-in flexibility, so that based on the pace of the class, and student engagement the schedule can be altered within a limit. Plan also includes time to support students with relatively lower attainment level.

In the module delivery we addressed role playing and communication. Also, as different groups choose different problems, students can experience different situations during the module delivery.

In the next section we reflect on our experience of running the module for past three years.

4. Discussion

We have run the module for three years and are preparing for the fourth cohort in 2024-25 academic year. The module is well received by the students from the inception of the module and gets good feedback in module review.

In addition to University's regular module feedback, we gather specific feedback each year and enhance the module content, delivery and sequence of delivery based on that feedback. For example, the module was offered to B.Engg (Software Engineering) students in the first year, later it was offered to B.Sc (Computer Science) and B.Sc (Computer Science with AI) students also based on student reception.

In the first instance, assessment was a combination of 70% course work and 30% examination. Based on student feedback, and in alignment with the aim of the module, we removed the examination and transformed it as 100% course work. The coursework is extensive to assess all learning outcomes.

We use the following questionnaire to gather student feedback every year. Except for the last question, all others are yes/no/not applicable questions.

- Did you learn anything new?
- Did you like the module delivery mechanism?
- Did you have to do a lot of work outside the classroom for your assignment?
- Could you relate the topics with your placement experience?
- Do you think it will help you in your job?
- Do you regret taking this module?
- Additional comments (if any)

Overall, we enjoy running this module and are learning a lot. We have learnt a few lessons. Firstly, for designing a new module, it is always advisable to judge how and where students will

 doi:10.20944/preprints202410.1690.v1

apply the knowledge acquired in the module. Secondly, students appreciate experiencing techniques applied in the real world. Being able to use enterprise design thinking or learning about software engineering practices leveraging IBM garage method was well appreciated. Finally, students enjoy working with problems that they can relate to their day-to-day life and are not much discussed in textbooks or learning materials. The problems presented in this module create a positive environment. It is also important to have an informal classroom environment to support learning.

However, the strength of the module is one of its biggest weaknesses also. The module delivery is fully dependent on in-class engagement of students, attendance of students had a major role to achieve the learning. Due to unavoidable circumstances (e.g., self-isolation due to proximity of covid patient, job interview, horrible weather condition, etc.) we always do not get desired attendance from all students. To mitigate this, we have designed the delivery schedule in a flexible way, keeping buffer time to allow additional support sessions or catch-up if required.

Next section concludes the paper.

3. Conclusion

In this paper we presented our experience of designing and delivering a new module 'Contemporary Software Engineering' to enhance employment skills for students.

We followed a structured process in analyzing the gaps between education and employment skills requirements for software engineers. We referred to literature and our experience engaging with local SME companies. We also analyzed the kind of organizations our students get engaged with. From our analysis, we prioritized addressing 'requirement & problem solving', 'architecture', 'software engineering methos & development process', 'quality', 'role playing' and 'communication' through the design and delivery of the module.

We discussed the design and delivery of the module and reflected upon our experience of running the module for three years. The major objective of this module was to enhance students' skills for employment and help easy transition to job. Student feedback indicates the objective is met. However, there is scope for improvements.

As part of future work, we will engage with the past students of the module to explore how well the module catered to their employment needs and how much of the topics delivered they are using in their employment. We will also continue our gap analysis between software engineering education & employment skills. We will keep on enhancing the design and delivery of the module based on newly identified gaps

References

1. V. Garousi, G. Giray, E. Tuzun, C. Catal and M. Felderer, "Closing the Gap Between Software Engineering Education and Industrial Needs," in IEEE Software, vol. 37, no. 2, pp. 68-77, March-April 2020, doi: 10.1109/MS.2018.2880823
2. Mary Shaw. 2000. Software engineering education: a roadmap. In ICSE-Future of SE Track. 371–380
3. Andrew Begel and Beth Simon. 2008. Struggles of new college graduates in their first software development job. SIGCSE Bull. 40, 1 (March 2008), 226–230. https://doi.org/10.1145/1352322.1352218
4. Alex Radermacher and Gursimran Walia. 2013. Gaps between industry expectations and the abilities of graduates. In Proceeding of the 44th ACM technical symposium on Computer science education (SIGCSE '13). Association for Computing Machinery, New York, NY, USA, 525–530 https://doi.org/10.1145/2445196.2445351
5. Alex Radermacher, Gursimran Walia, and Dean Knudson. 2014. Investigating the skill gap between graduating students and industry expectations. In Companion Proceedings of the 36th International Conference on Software Engineering (ICSE Companion 2014). Association for Computing Machinery, New York, NY, USA, 291–300. https://doi.org/10.1145/2591062.2591159
6. Nigel Shadbolt. 2016. Shadbolt review of computer sciences degree accreditation and graduate employability: April 2016. (2016). https://dera.ioe.ac.uk/16232/2/ind-16-5-shadbolt-review-computer-science-graduate-employability_Redacted.pdf
7. Joseph Maguire, Steve Draper, and Quintin Cutts. 2019. What Do We Do When We Teach Software Engineering? In Proceedings of the 2019 Conference on United Kingdom & Ireland Computing Education Research (UKICER '19). Association for Computing Machinery, New York, NY, USA, Article 10, 1–7 https://doi.org/10.1145/3351287.3351295

 doi:10.20944/preprints202410.1690.v1

8. Melegati, J., Kon, F. (2020). Early-Stage Software Startups: Main Challenges and Possible Answers. In: Nguyen-Duc, A., Münch, J., Prikladnicki, R., Wang, X., Abrahamsson, P. (eds) Fundamentals of Software Startups. Springer, Cham. https://doi.org/10.1007/978-3-030-35983-6_8
9. E Ries.: The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. Crown Business, New York (2011)
10. Melegati, J., Chanin, R., Wang, X., Sales, A., Prikladnicki, R. (2019). Enablers and Inhibitors of Experimentation in Early-Stage Software Startups. In: Franch, X., Männistö, T., Martínez-Fernández, S. (eds) Product-Focused Software Process Improvement. PROFES 2019. Lecture Notes in Computer Science(), vol 11915. Springer, Cham. https://doi.org/10.1007/978-3-030-35333-9_39
11. Gupta, V., Hoy, Z., & Gupta, C. (2022). Empirical insights into software startups. In V. Gupta, & C. Gupta (Eds.), Emerging Technologies for Innovation Management in the Software Industry (pp. 151-156). IGI Global. https://doi.org/10.4018/978-1-7998-9059-1.ch008
12. Islam, M. A., & Alghobiri, M. A. (2019). E-Entrepreneurship for E-Startups: Potentials, Common Challenges and Way Forward. Information Management and Business Review, 10(4), 44-50. https://doi.org/10.22610/imbr.v10i4.2646
13. Knowledge transfer partnership https://www.ktp-uk.org/ last accessed on 10th October 2024
14. Cheryl Nakata, Jiyoung Hwang, Design thinking for innovation: Composition, consequence, and contingency, Journal of Business Research, Volume 118, 2020, Pages 117-128, ISSN 0148-2963, https://doi.org/10.1016/j.jbusres.2020.06.038
15. Da Silva, Ricardo Henrique; Kaminski, Paulo C; Armellini, Fabiano Improving new product development innovation effectiveness by using problem solving tools during the conceptual development phase: Integrating Design Thinking and TRIZ ISSN: 0963-1690 , 1467-8691; Creativity and innovation management. , 2020, Vol.29(4), p.685-70 https://doi.org/10.1111/caim.12399
16. Enterprise Design Thinking IBM https://www.ibm.com/design/approach/design-thinking/ last accessed on 10th October 2024
17. Gerrit Muller A Reference Architecture Primer University of South-Eastern Norway-NISE available at https://gaudisite.nl/ReferenceArchitecturePrimerSlides.pdf last accessed on 11th October 2022
18. R. Akbar, "Tailoring Agile-Based Software Development Processes," in IEEE Access, vol. 7, pp. 139852-139869, 2019, doi: 10.1109/ACCESS.2019.2944122
19. IBM Garage Method https://www.ibm.com/garage last accessed on 10th October 2024
20. E Ries.: The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. Crown Business, New York (2011)
21. ISO 9000 family https://www.iso.org/iso-9001-quality-management.html last accessed on 10th October 2024
22. Richard C. Linger,Mark C. Paulk, Carmen J. Trammell, Cleanroom Software Engineering Implementation of the Capability Maturity Model (CMMsm) for Software, A report from Software Engineering Institute, Carnegie Mellon University, Pittsburgh available at https://resources.sei.cmu.edu/asset_files/TechnicalReport/1996_005_001_16505.pdf last accessed on 10th October 2024