CSC585: Assignment-3

Submitted by: **Malika Hafiza Pasha**

Question 1: The evaluation of project product conformance requires evaluating software products and the related documentation.

a. **Explain why evaluation of software product alone (the product directly serving the user) is not sufficient.**
b. **Provide at least two situations where missing documentation or documentation not evaluated could cause difficulties in the development process and in the maintenance of the software product.**

Answer:

Evaluating a software product's conformance is a multifaceted process that extends beyond the examination of the product itself to include its related documentation. This comprehensive approach is essential for several reasons, which I will elaborate on in response to your questions.

a. Why Evaluation of Software Product Alone is Not Sufficient

i. **Complexity and Usability:**
Comprehensive documentation can help alleviate the inherent complexity of software systems, particularly those intended for complicated tasks. Users and developers can get clarification on the features, functionalities, and usage of the product through the documentation, which acts as a guide. It becomes difficult to use the software to its maximum capacity without it.

ii. **Maintenance and Support:**
Software products need to be updated, maintained, and occasionally debugged over time. Enough documentation guarantees that fresh developers or support teams can comprehend the architecture and operational logic of the product. This includes code comments, API documentation, and maintenance manuals. This knowledge is essential for effective updates, improvements, and problem solving.

iii. **Scalability and Enhancements:**
Software products frequently need to be improved or scaled as firms expand. By describing the software's present features, constraints, and architecture, documentation plays a crucial part in this process and helps decision-makers make well-informed choices about scalability and improvements.

iv. **Regulatory Compliance and Auditing:**
Meeting compliance standards is essential for many software products, particularly those in regulated businesses (such as finance, healthcare, and aviation). Documentation explains how

the program complies with regulatory requirements, which helps with auditing procedures and offers proof of compliance.

b. Situations Where Missing Documentation or Unassessed Documentation Could Cause Difficulties

i. **Onboarding New Team Members:**
New team members rely on documentation during the development process to acquaint themselves with the features, architecture, and coding standards of the software product. The learning curve can be further accelerated by inadequate documentation, which can cause delays and higher onboarding expenses. Furthermore, if there is inadequate documentation, there is a greater chance of miscommunications and mistakes, which could lower the program's quality.

ii. **Maintaining and Updating the Software:**
Updates, feature additions, and bug patches are all part of software maintenance. Developers may find it difficult to comprehend the reasoning behind specific design choices, dependencies, and the architecture in the absence of thorough documentation. This ignorance might result in ineffective maintenance procedures as developers are overly preoccupied with reading the codebase instead of putting fixes in place. Undocumented code or features also present a serious update risk since modifications made to one area of the program may unintentionally impact other areas, creating new vulnerabilities and issues.

In conclusion, because of the complexity of its operation, upkeep, and improvement, assessing a software product's conformity based only on the product itself is inadequate. Ensuring usability, supporting scalability, facilitating maintenance, and adhering to legal standards all depend on thorough documentation. The absence or insufficient assessment of documentation can pose considerable difficulties during the software development and maintenance phases, ultimately affecting the software product's longevity and quality.

Question 2: Some people claim that one of the justifications for a small design review team is the need to schedule the review session within a few days after the design product has been distributed to the team members.

a. List additional reasons for preferring small DR teams, apart from the anticipated delays in convening a DR session composed of large teams.

b. What reasons motivate attempts to schedule the review session as soon after distribution of the design report to the team members as possible?

Answer:

Design reviews are crucial steps in the development process of products, systems, or software, aiming to ensure that the design meets the required specifications and standards before moving forward. While the ease of scheduling is one important reason for preferring small design review (DR) teams, there are several other considerations that motivate this preference:

a. Additional Reasons for Preferring Small DR Teams:

i. **Increased Efficiency and Focus:**
During conversations, smaller teams are typically more productive and focused. More chances exist for each participant to contribute, and the discussion can stay narrowly focused on important subjects without straying into unrelated subjects.

ii. **Better Communication:**
In small teams, there is usually more direct and efficient communication. Misunderstandings are less common and questions and concerns can be resolved faster when there are fewer individuals present.

iii. **Enhanced Accountability:**
Assigning and monitoring tasks is simpler with small teams. The increased visibility of each member's efforts heightens their sense of responsibility and dedication to the project's success.

iv. **Faster Decision-Making:**
Decisions can be made faster when there are fewer participants. In larger organizations, the requirement to take into account and include more viewpoints might slow down the process of coming to a decision or establishing consensus.

v. **Decreased Social Loafing:**
In smaller teams, social loafing—the phenomenon where people work less hard to accomplish a task when they collaborate with others than when they work alone—is less of a problem. Because everyone's contribution is essential to the result, there is less likelihood that members will rely on one another to do the heavy lifting.

vi. **Cost-Effectiveness:**
When outside experts are involved, small teams may be more economical. Reduced staffing means cheaper review-related costs, like airfare, lodging, and time away from other duties.

b. Reasons to Schedule the Review Session Soon After Distribution of the Design Report:

i. **Freshness of Material:**
Scheduling the review session shortly after distribution guarantees that the reviewers' memories of the material are current. This may result in feedback that is more informative and lessen the need for lengthy recaps.

ii. Maintaining Momentum:

Prompt scheduling aids in keeping the project moving forward. Prolonged delays may cause people to lose interest in the project or to focus on other things, which could cause the timeline to go out of control.

iii. **Rapid Identification of Issues:**
Early assessment makes it possible to quickly identify any significant problems or weaknesses in the design, giving more time for changes and enhancements prior to further development.

iv. **Resource Allocation:**
By spotting design flaws early on, resources (money, manpower, and time) may be redirected more effectively, keeping the project moving forward.

v. **Stakeholder Satisfaction:**
By showcasing accomplishments and guaranteeing that their issues are promptly resolved, prompt reviews can raise stakeholder satisfaction. Maintaining support for the project may depend on this.

vi. **Regulatory Compliance:**
Early evaluations can help ensure that the design conforms with all relevant standards and rules for projects in regulated industries, preventing expensive adjustments later on.

In conclusion, small design review teams provide many advantages for increased productivity, communication, and overall efficacy of the review process, in addition to making scheduling simpler. Reviewing the materials soon after they are distributed ensures that the discussions are pertinent, keeps the project moving forward, and permits fast revisions in response to input.

Question 3: "In most cases, the test case file preferred should combine sample cases with synthetic cases, to overcome the disadvantages of a single source of test cases, and to increase the efficiency of the testing process" (See Section 14.5.1).

a. **Elaborate on how applying a mixed-source methodology overcomes the disadvantages of a single-source methodology.**
b. **Elaborate on how applying a mixed-source methodology enhances testing efficiency. Provide a hypothetical example.**

Answer:

a. Elaborating on how to apply a mixed-source methodology overcoming the disadvantages of a single-source methodology:

Software testing that uses a mixed-source approach, which mixes synthetic and sample cases, attempts to produce a test suite that encompasses a wide range of plausible scenarios that are not achievable with single-source approaches alone. Real-world sample cases have the advantage of being able to replicate the data and usage that the software would actually experience in production, which includes a combination of edge cases and average examples based on real user behavior. They might not, however, cover every circumstance, particularly the uncommon or unexpected ones, and they might overly repeat typical cases, which would be ineffective.

However, well-crafted synthetic test cases might target particular situations and edge cases that might not have been included in real-world samples but are theoretically feasible within the constraints of the software. They can guarantee coverage of places that might otherwise go unnoticed by doing this.

By combining the natural variety and realism of sample instances with the comprehensiveness and emphasis of synthetic examples, the mixed-source approach lessens these drawbacks. This combination increases the possibility of finding both common and uncommon errors by enabling testers to cover a wide range of test scenarios.

b. Elaborating on how to apply a mixed-source methodology enhances testing efficiency:

The mixed-source methodology improves software testing efficiency by enabling a more efficient test suite that minimizes redundant efforts and concentrates on optimizing coverage through a well-balanced set of cases. The technique makes sure that both often occurring scenarios and probable edge cases are checked, which raises the likelihood of finding problems.

Hypothetical Example:

Consider a piece of software that a bank uses to handle loan applications. If testing is restricted to the most typical loan application types, it may not test the software's ability to handle less common but still potential circumstances, like applications with non-standard property kinds or atypical income sources. However, depending solely on synthetic examples could result in an overemphasis on edge cases, which are significant but do not represent most real-world inputs.

The testing team can use actual data to cover normal loan processing scenarios—which account for the majority of the bank's loan applications—by utilizing a mixed-source methodology. Next, in order to verify the software's behavior, they can add synthetic cases that simulate difficult or unusual loan scenarios, including loans involving applicants with numerous income streams or those requiring foreign credit checks.

By using this technique, it is ensured that the software is tested against both the most common user situations and the edge cases that, if not handled properly, could result in serious problems. Because it has been tested against a wide range of scenarios that are reflective of both common and uncommon conditions, the software program is more resilient and reliable when it is released.

Question 4: Most software operation procedures require extensive documentation in the activities

performed.

a. **List the main uses for the various types of software operation documentation.**
b. **Explain the importance of the required documentation in your own words?**

Answer:

An essential component of guaranteeing the seamless and effective use and upkeep of software programs is software operation documentation. Below is a summary of the primary applications for different kinds of software operation documentation along with the significance of having this documentation.

a. Main Uses for Various Types of Software Operation Documentation:

i. **User Manuals:**
The purpose of these publications is to aid end users in comprehending how to operate the software. They frequently offer troubleshooting advice, feature explanations, and step-by-step instructions. Software adoption and the mitigation of the learning curve that comes with new software depend on user guides.

ii. **System Administration Guides:**
IT specialists who are in charge of installing, configuring, and maintaining the software are the intended audience for these publications. In-depth technical details about system requirements, installation methods, configuration options, and system upkeep are all included. The proper deployment and continued functionality of the software are guaranteed by this kind of documentation.

iii. **API Documentation:**
It is imperative that software that provides an Application Programming Interface (API) has this documentation. Endpoints, data formats, and sample code are among the things it gives developers so they may learn how to communicate with the software programmatically. When creating bespoke apps or integrating the program with other systems, API documentation is crucial.

iv. **Release Notes:**
These documents contain details on all of the software's updates, including patches for bugs, enhancements, and new features. It's critical that administrators and users read the release notes to learn about any changes and how they may affect how they use the program.

v. **Troubleshooting Guides:**
These guidelines, which include diagnostic data and answers to known problems, are intended to assist users and administrators in resolving typical issues. They are essential for decreasing downtime and raising user contentment.

vi. **Operational Procedures:**
The normal operating procedures for regular tasks, backups, recovery, and performance tuning are described in these publications. They guarantee regularity in operations and facilitate prompt recovery in the event of malfunctions or calamities.

b. The Importance of Required Documentation:

The importance of software operation documentation can be understood from several perspectives:

i. **Efficiency and Productivity:**
Effective documentation serves as a self-service tool for administrators and users, enabling them to comprehend features and fix problems without outside assistance. Because more time is spent productively and less time is wasted on troubleshooting, this results in increased efficiency and production.

ii. **Quality and Reliability:**
The software's general dependability and quality are enhanced by the documentation. By ensuring that the program is used as intended and that any problems are promptly fixed, it makes the system more dependable and stable.

iii. **Knowledge Transfer:**
Documentation serves as a key tool for knowledge transfer. It makes ensuring that when people change jobs or leave an organization, their understanding of how to operate and maintain the software is retained. The preservation of operational stability depends on this continuity.

iv. **Compliance and Security:**
Several organizations are required by law to maintain thorough documentation. It guarantees that security mechanisms are appropriately implemented and adhered to, and that the software complies with industry standards.

v. **Customer Satisfaction:**
Lastly, satisfied customers are directly impacted by well-written documentation. It is more probable that users would enjoy using the program if they can quickly find solutions to their issues and answers to their inquiries.

In conclusion, the successful, dependable, and efficient usage of software depends on the existence of software operating documentation. It helps users and administrators, guarantees security and compliance, promotes knowledge transfer, and eventually raises end-user happiness.

Question 5: A planned human resource software system is estimated to require 5,000 logical statements of MYSQL code.

a. Estimate the number of function points required for the software system.

b. Estimate the number of months required for a team of three members to complete the software system.

Answer:

The software system's function points must be estimated, and we also need to project how long it will take a three-person team to finish the system.

Part a: Estimate the Number of Function Points

A common metric for estimating the size of a software development project that accounts for several facets of the program's functionality is called Function Points (FP). However, because function points are based more on the functionality being provided than the lines of code or statements stated, translating logical statements of SQL code directly into function points is not an easy task.

However, if we were to tackle this, we would take into account that the nature of SQL statements frequently corresponds with data manipulation and retrieval operations, which in the context of Function Point Analysis could be categorized under External Inputs (EI), External Outputs (EO), and External Inquiries (EQ). Every one of these kinds has a unique function point weight.

When there is no direct conversion ratio available, industry average productivity rates—which vary according on the language and complexity of the code—can occasionally be used to approximate the number of source lines of code (SLOC) per FP. Since SQL is a specialized domain language, it may not fit neatly into other software language categories. However, let's suppose for estimation purposes that 50–100 SLOC (SQL in this example) is equal to one FP on average.

If we take each of the 5,000 logical SQL statements to be one SLOC, we get 5,000 SLOC.

Using 75 SLOC per FP as the midpoint of the 50-100 range for the average conversion rate:

Function Points = (5000 SLOC) / (75 SLOC/FP)
It is predicted that the software system will require roughly 67 function points.

Part b: Estimate the Number of Months Required

We can utilize the Function Points to calculate effort and divide that result by the team's capacity to estimate the development time. Person-Months per Function Point is a frequently used measure that varies based on the team's experience, the development environment, and the project's complexity. For a project of modest complexity, an industry average may be approximately 0.1 to 0.2 person-months per FP.

Let's calculate the effort first:

Effort (Person-Months) = Function Points × Person-Months/FP

Given a team of 3 members, the time to complete in months would be:

Time to Complete (Months) = Effort (Person-Months) / Team Size

A three-person team should be able to finish the software system in between 2.2 and 4.4 months on average. This range considers a moderate level of project complexity, with lower and higher ranges that represent variations in project specifics and production rates.

These approximations function as a general guideline. The precise needs of the software system, the team's familiarity with SQL and related technologies, and the effectiveness of their development methods are just a few examples of the variables that can greatly affect the actual results.

Question 6: The table of control suggested in Frame 17.2 (of the textbook) includes an optional section, "Terms and Abbreviations."

a. Do you recommend including terms like software program, printed output, configuration management, or ATM in this section? List your arguments.

b. What criteria should be applied when including a term or abbreviation? List your reasoning.

Answer:

a. Incorporating certain terms such as "printed output," "software program," "configuration management," or "ATM" into the "Terms and Abbreviations" section may prove advantageous. The following justifies the inclusion of these terms:

i. Clarity and Precision:
By providing definitions for these terms, you can make sure that every reader knows exactly what is intended by them in relation to the procedures that are being explained. This is especially crucial if a broad readership with differing degrees of experience or backgrounds in the workplace is the intended audience for the paper.

ii. Consistency:

By allowing terms and abbreviations to be referred to, a section on terms and abbreviations can assist guarantee that terminology are used consistently across the document.

iii. Complexity of Terms:

In the context of software development and quality assurance, terms like "configuration management" may have distinct meanings from their everyday usage. Giving a definition aid in preventing confusion.

iv. **Industry-Specific Jargon:**

A lot of acronyms and jargon used in the software industry may not be recognizable to all readers. By providing definitions, we make sure that everyone, including those who are not experts in the industry, can comprehend the processes.

v. **Preventing Ambiguity:**

While some terminology, like "ATM," can refer to various things (automated teller machine, asynchronous transfer mode), they should be avoided. By adding them to the section, the intended use will become clearer.

b. The criteria for including a term or abbreviation should be as follows:

i. **Relevance:**
The phrase or acronym ought to have a clear connection to the processes' subject matter. It has to be something that is essential to comprehending the procedures and is probably going to be used in the text.

ii. **Frequency of Use:**
To improve comprehension, terms and abbreviations that are used frequently throughout the document should be provided.

iii. **Potential for Misinterpretation:**
A term should be included and defined precisely if there is a chance that it may be misinterpreted or if it is frequently misinterpreted in relation to the subject matter.

iv. **Specialized Use:**
Terms and acronyms that, in the context of the procedures, have a distinct or specialized meaning that may vary from their general use, should be provided.

v. **Audience Knowledge Level:**
Take into account the audience's level of knowledge. Include a term if you think the intended readers of the paper would not be familiar with it.

vi. **Industry Standards:**
To conform to the common industry vocabulary, a phrase or abbreviation that is standardized and widely accepted in the industry should be included.

In conclusion, the necessity of a phrase or abbreviation for comprehending the document, the possibility of a misunderstanding, and the requirements of the intended audience should all be taken into consideration when deciding whether or not to include it. Terms and acronyms that are essential to the document's accuracy and clarity should always be used.