

University of Sargodha

BS 3rd Term Examination 2016

Subject: Software Engineering

Paper: Software Engineering (CMP: 3310)

Q1. Write short answers of the following.

i. What is CMM?

Capability Maturity Model (CMM) is a technical and cross-discipline methodology used to facilitate and refine software development processes and system improvement. Based on the Process Maturity Framework (PMF), CMM was developed to assess the performance capabilities of government contractors. Its levels are as follows:

- 1) Initial
- 2) Repeatable/Managed
- 3) Defined
- 4) Quantitatively Managed
- 5) Optimizing

ii. What is System Testing?

System Testing is the testing of a complete and fully integrated software product. Usually software is only one element of a larger computer based system. Ultimately, software is interfaced with other software/hardware systems. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer based system. System test falls under the black box testing category of software testing.

iii. Differentiate between Project, Process and Product?

A software process specifies a method of development software. A software project, on the other hand is a development project in which a software process is used. And software products are the outcomes of a software project.

Each software development project starts with some needs and (hopefully) ends with some software that satisfies those needs. A software process specifies the abstract set of activities that should be performed to go from user needs to final product. The actual act of executing the activities for some specific user needs is a software project. And all the outputs that are produced while the activities are being executed are the products.

iv. Define Stakeholder?

A stakeholder is a person or group who has an interest -- vested or otherwise -- in an enterprise and whose support is required in order for an enterprise to be successful. A stakeholder is a party that has an interest in a company and can either affect or be affected by the business. The primary stakeholders in a typical corporation are its investors, employees, customers and suppliers.

v. What is SWOT Analysis?

SWOT analysis (strengths, weaknesses, opportunities and threats analysis) is a framework for identifying and analyzing the internal and external factors that can have an impact on the viability of a project, product, place or person.

SWOT analysis is most commonly used by business entities, but it is also used by nonprofit organizations and, to a lesser degree, individuals for personal assessment. Additionally, it can be used to assess initiatives, products or projects.

vi. What is Software Refactoring?

Refactoring is the process of altering an application's source code without changing its external behavior. The purpose of code refactoring is to improve some of the nonfunctional properties of the code, such as readability, complexity, maintainability and extensibility.

Refactoring can extend the life of source code, preventing it from becoming legacy code. The refactoring process makes future enhancements to such code a more pleasant experience. Refactoring is also known as reengineering.

vii. What is Embedded Systems?

An embedded system is a dedicated computer system designed for one or two specific functions. This system is embedded as a part of a complete device system that includes hardware, such as electrical and mechanical components.

Because an embedded system is engineered to perform certain tasks only, design engineers may optimize size, cost, power consumption, reliability and performance.

viii. What's the difference between Cohesion and Coupling?

| Cohesion | Coupling |
|--|--|
| Cohesion is the indication of the relationship within module. | Coupling is the indication of the relationships between modules. |
| Cohesion shows the module's relative functional strength. | Coupling shows the relative independence among the modules. |
| Cohesion is a degree (quality) to which a component / module focuses on the single thing. | Coupling is a degree to which a component / module is connected to the other modules. |
| While designing you should strive for high cohesion i.e. a cohesive component/ module focus on a single task (i.e., single-mindedness) with little interaction with other modules of the system. | While designing you should strive for low coupling i.e. dependency between modules should be less. |

ix. Define SQA?

Software quality assurance (SQA) is a process that ensures that developed software meets and complies with defined or standardized quality specifications. SQA is an ongoing process within the software development life cycle (SDLC) that routinely checks the developed software to ensure it meets desired quality measures.

x. What is Gantt Chart?

A Gantt chart is a horizontal bar chart developed as a production control tool in 1917 by Henry L. Gantt, an American engineer and social scientist. Frequently used in project management, a Gantt chart provides a graphical illustration of a schedule that helps to plan, coordinate, and track specific tasks in a project.

xi. Write down the name of different phases of Rational Unified Process?

The four phases of Rational Unified Process are as follows:

- 1) Inception
- 2) Elaboration
- 3) Construction
- 4) Transition

xii. What's the difference between Testing and Debugging?

| Testing | Debugging |
|--|---|
| The purpose of testing is to find bugs and errors. | The purpose of debugging is to correct those bugs found during testing. |
| Testing is done by tester. | Debugging is done by programmer or developer. |
| It can be automated. | It can't be automated. |
| It can be done by outsider like client. | It must be done only by insider i.e. programmer. |

| | |
|---|--|
| Most of the testing can be done without design knowledge. | Debugging can't be done without proper design knowledge. |
|---|--|

xiii. Differentiate between Iterative and Incremental Model?

The Incremental approach is a method of software development where the model is designed, implemented and tested incrementally (a little more is added each time) until the product is finished. It involves both development and maintenance. The product is defined as finished when it satisfies all of its requirements

The Iterative Design is a design methodology based on a cyclic process of prototyping, testing, analyzing, and refining a product or process. Based on the results of testing the most recent iteration of a design, changes and refinements are made. This process is intended to ultimately improve the quality and functionality of a design.

xiv. Define Preliminary Investigation?

Preliminary investigation is the first phase. In this phase, the system is investigated. The objective of this phase is to conduct an initial analysis and findings of the system as follows:

- (1). **System Identification:** This phase is used to identify the need for a new system. It is very important step. Everything performed in future depends on this phase.
- (2). **System Scope:** The scope of the system is defined at this stage. The scope of a system can be reduced due to financial and political problems etc.
- (3). **Feasibility Study:** A system must be affordable and acceptable for the organization. A feasibility study is used to find whether the proposed system is feasible. This decision depends on different issues such as financial and time limits.
- (4). **Preliminary Plan:** Preliminary plan consists of all findings in written form for approval. It is also called feasibility report. It is normally submitted to the top managers of the organization. They may accept, modify or reject the report.

xv. What's the difference between Data and Process Modeling?

Data model represents the data objects and the interactions among the data objects in an organization, while the process model is a diagrammatic representation of a sequence of activities in an organization. The data model could be seen as a part of the business process model, which specifies how the information in the organization should be stored effectively to improve the overall performance. In a typical organization there are important interactions between the data model and the business process model.

xvi. What's the Integration Testing?

Integration testing, also known as integration and testing (I&T), is a software development process which program units are combined and tested as groups in multiple ways. In this context, a unit is defined as the smallest testable part of an application. Integration testing can expose problems with the interfaces among program components before trouble occurs in real-world program execution. Integration testing is a component of Extreme Programming (XP).

Subjective Part

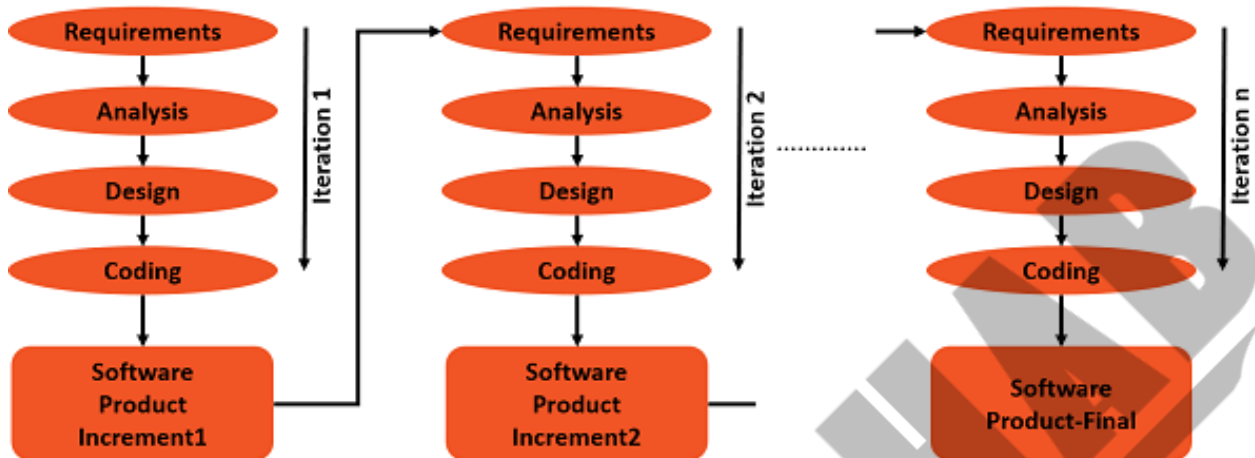
Q2. Compare and Contrast between Incremental Process Model and Evolutionary Process Model?

➤ Incremental Process Model

The incremental model delivers a series of releases, called increments that provide progressively more functionality for the customer as each increment is delivered. The incremental model combines elements of linear and parallel process flows (at page 13).

“Incremental Model is a process of software development where requirements are broken down into multiple standalone modules of software development cycle. Incremental development is done in steps from analysis design, implementation, testing/verification, maintenance.”

When an incremental model is used, the first increment is often a core product. That is, basic requirements are addressed but many supplementary features (some known, others unknown) remain undelivered. The core product is used by the customer (or undergoes detailed evaluation). As a result of use and/or evaluation, a plan is developed for the next increment. The plan addresses the modification of the core product to better meet the needs of the customer and the delivery of additional features and functionality. This process is repeated following the delivery of each increment, until the complete product is produced.



➤ Advantages of Incremental Process Model

The advantages or strengths of Iterative Incremental model are:

- ✓ You can develop prioritized requirements first.
- ✓ Initial product delivery is faster.
- ✓ Customers gets important functionality early.
- ✓ Lowers initial delivery cost.
- ✓ Each release is a product increment, so that the customer will have a working product at hand all the time.
- ✓ Customer can provide feedback to each product increment, thus avoiding surprises at the end of development.
- ✓ Requirements changes can be easily accommodated.

➤ Disadvantages of Incremental Process Model

The disadvantages of the Iterative Incremental model are:

- ✓ Requires effective planning of iterations.
- ✓ Requires efficient design to ensure inclusion of the required functionality and provision for changes later.
- ✓ Requires early definition of a complete and fully functional system to allow the definition of increments.
- ✓ Well-defined module interfaces are required, as some are developed long before others are developed.
- ✓ Total cost of the complete system is not lower.

When to Use Incremental Process Model?

Iterative Incremental model can be used when:

- ✓ Most of the requirements are known up-front but are expected to evolve over time.
- ✓ The requirements are prioritized.
- ✓ There is a need to get the basic functionality delivered fast.
- ✓ A project has lengthy development schedules.

Solved by Talha Shahab

- ✓ A project has new technology.
- ✓ The domain is new to the team.

➤ Evolutionary Process Models

Evolutionary process models produce an increasingly more complete version of the software with each iteration. Evolutionary models are iterative type models. They allow to develop more complete versions of the software.

Following are the evolutionary process models.

- The prototyping model**
- The spiral model**
- Concurrent development model**

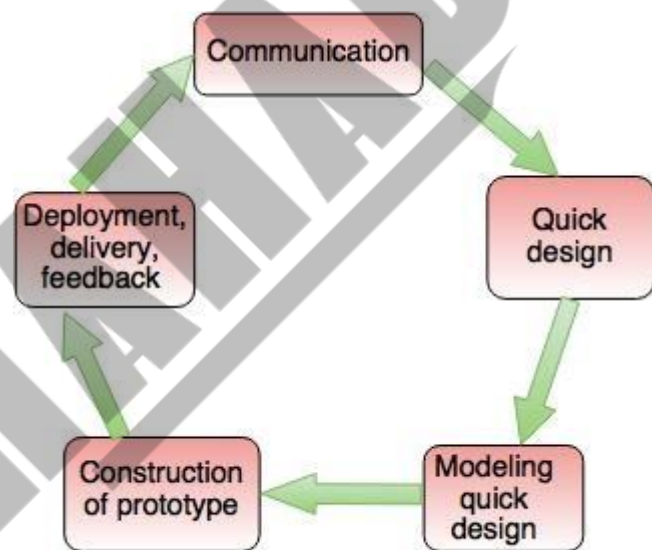
Their details are as follows:

A. The Prototyping model

Prototype is defined as first or preliminary form using which other forms are copied or derived. Prototype model is a set of general objectives for software. It does not identify the requirements like detailed input, output. It is software working model of limited functionality. In this model, working programs are quickly produced.

The different phases of Prototyping model are:

1. **Communication:** In this phase, developer and customer meet and discuss the overall objectives of the software.
2. **Quick design:** Quick design is implemented when requirements are known. It includes only the important aspects like input and output format of the software. It focuses on those aspects which are visible to the user rather than the detailed plan. It helps to construct a prototype.
3. **Modeling quick design:** This phase gives the clear idea about the development of software because the software is now built. It allows the developer to better understand the exact requirements.
4. **Construction of prototype:** The prototype is evaluated by the customer itself.
5. **Deployment, delivery, feedback:** If the user is not satisfied with current prototype then it refines according to the requirements of the user. The process of refining the prototype is repeated until all the requirements of users are met. When the users are satisfied with the developed prototype then the system is developed on the basis of final prototype.



Advantages of Prototyping Model

- ✓ Prototype model need not know the detailed input, output, processes, adaptability of operating system and full machine interaction.
- ✓ In the development process of this model users are actively involved.
- ✓ The development process is the best platform to understand the system by the user.
- ✓ Errors are detected much earlier.
- ✓ Gives quick user feedback for better solutions.
- ✓ It identifies the missing functionality easily. It also identifies the confusing or difficult functions.

Disadvantages of Prototyping Model:

- ✓ The client involvement is more and it is not always considered by the developer.
- ✓ It is a slow process because it takes more time for development.

- ✓ Many changes can disturb the rhythm of the development team.
- ✓ It is a thrown away prototype when the users are confused with it.

Q3. Define Software Design? Explain all sort of design in detail.

*{Topic no. 6.4 and 6.5 (User Interface Design and WebApps Interface Design) on page no. 49, 50 in notes}
 {Page no. 317 and 335 in book "Software Engineering: A Practitioner's Approach 7th Edition by Roger S. Pressman"}*

Q4. A New Branch School District Operates a fleet of 40 buses that serves approximately 1,000 students in grade 1 to 12. The bus operation involves 30 regular routes, plus special routes for activities, athletic events and summer sessions. The district employs 12 fulltime drivers and 25 to 30 part-time drivers. A dispatcher coordinates the staffing and routes and relays messages to drivers regarding students and parents who call about pickup and drop-off arrangements.

- a. Identify possible actors and use cases involved in school bus operation.
- b. Create a use case diagram using any one of these use cases identified in part a.
- c. Create a sequence diagram for the use case you selected.

Answer:

a. Identify possible actors and use cases involved in school bus operation.

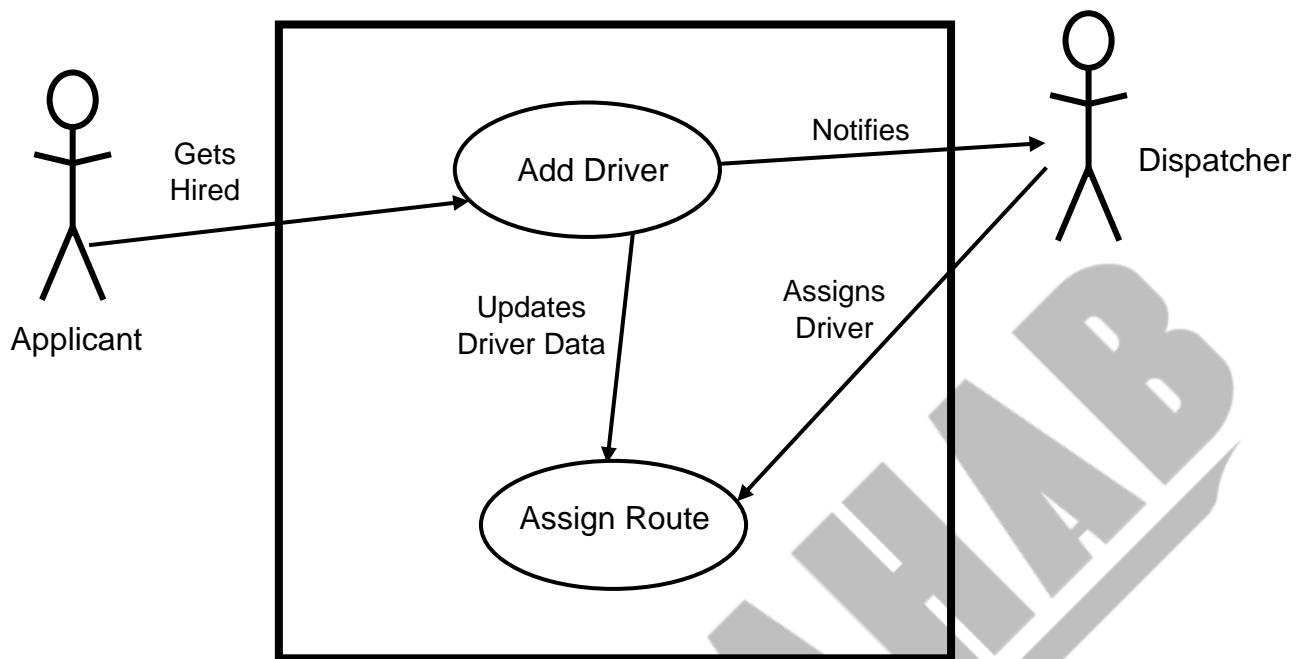
The actors in the above situation might include Applicant, Dispatcher, Driver, and Student. Use cases might include Add Driver, Add Route, Assign Student, and Assign Seat.

A sample use case diagram for the Add Driver use case follows:

| | |
|-------------------------------|---|
| Name: | Add Driver |
| Actor: | Dispatcher/Driver |
| Description: | This use case describes the process used to add a new bus driver assignment |
| Successful Completion: | Dispatcher adds driver to records Dispatcher assigns driver to available route Dispatcher notifies school, and driver of assignment Dispatcher gives keys to driver |
| Alternative: | Dispatcher adds driver to records Dispatcher assigns driver to substitute list Dispatcher notifies school, and driver of substitute assignment Dispatcher puts driver on waiting list for permanent route assignment |
| Pre-condition: | Applicant possesses appropriate license |
| Post-Condition: | Driver entered in records and given assignment |
| Assumptions: | None |

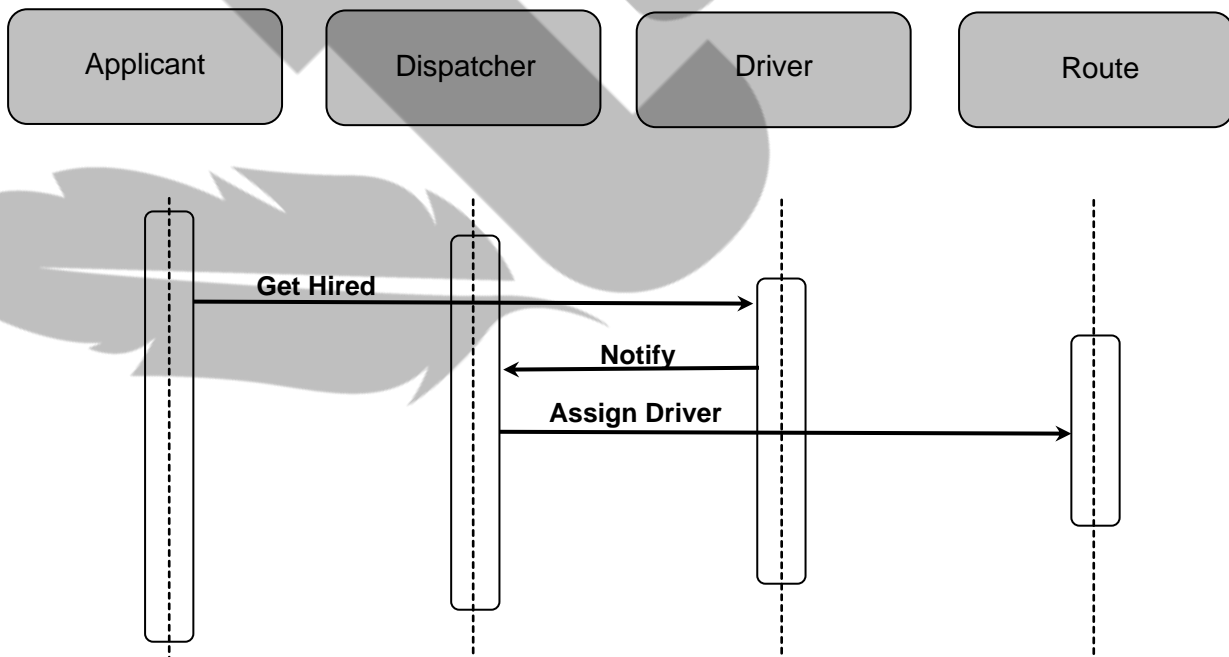
b. Create a use case diagram using any one of these use cases identified in part a.

A sample use case diagram for the Add Driver and Assign Route use cases follows:



c. Create a sequence diagram for the use case you selected.

A sample sequence diagram for the Add Driver use case follows:



Q5. Write a note on different types of Software Testing Techniques?

- ☒ Unit Testing
- ☒ Integration Testing
- ☒ Regression Testing

Solved by Talha Shahab

☑ Smoke Testing

There are many strategies that can be used to test software. At one extreme, you can wait until the system is fully constructed and then conduct tests on the overall system in hopes of finding errors. This approach, although appealing, simply does not work. It will result in buggy software that disappoints all stakeholders. At the other extreme, you could conduct tests on a daily basis, whenever any part of the system is constructed. This approach, although less appealing to many, can be very effective. Unfortunately, some software developers hesitate to use it.

A testing strategy that is chosen by most software teams falls between the two extremes. It takes an incremental view of testing, beginning with the testing of individual program units, moving to tests designed to facilitate the integration of the units, and culminating with tests that exercise the constructed system. Each of these classes of tests is described in the sections that follow.

🚦 Unit Testing

Unit testing focuses verification effort on the smallest unit of software design—the software component or module. Using the component-level design description as a guide, important control paths are tested to uncover errors within the boundary of the module. The relative complexity of tests and the errors those tests uncover is limited by the constrained scope established for unit testing. The unit test focuses on the internal processing logic and data structures within the boundaries of a component. This type of testing can be conducted in parallel for multiple components.

➤ Unit-test considerations.

Data flow across a component interface is tested before any other testing is initiated. If data do not enter and exit properly, all other tests are moot. In addition, local data structures should be exercised and the local impact on global data should be ascertained (if possible) during unit testing.

Selective testing of execution paths is an essential task during the unit test. Test cases should be designed to uncover errors due to erroneous computations, incorrect comparisons, or improper control flow.

Boundary testing is one of the most important unit testing tasks. Software often fails at its boundaries. That is, errors often occur when the n th element of an n -dimensional array is processed, when the i th repetition of a loop with i passes is invoked, when the maximum or minimum allowable value is encountered. Test cases that exercise data structure, control flow, and data values just below, at, and just above maxima and minima are very likely to uncover errors.

Among the potential errors that should be tested when error handling is evaluated are:

- (1) Error description is unintelligible,
- (2) Error noted does not correspond to error encountered,
- (3) Error condition causes system intervention prior to error handling,
- (4) Exception-condition processing is incorrect,
- (5) Error description does not provide enough information to assist in the location of the cause of the error.

➤ Unit-test procedures.

Unit testing is normally considered as an adjunct to the coding step. The design of unit tests can occur before coding begins or after source code has been generated. A review of design information provides guidance for establishing test cases that are likely to uncover errors in each of the categories discussed earlier. Each test case should be coupled with a set of expected results.

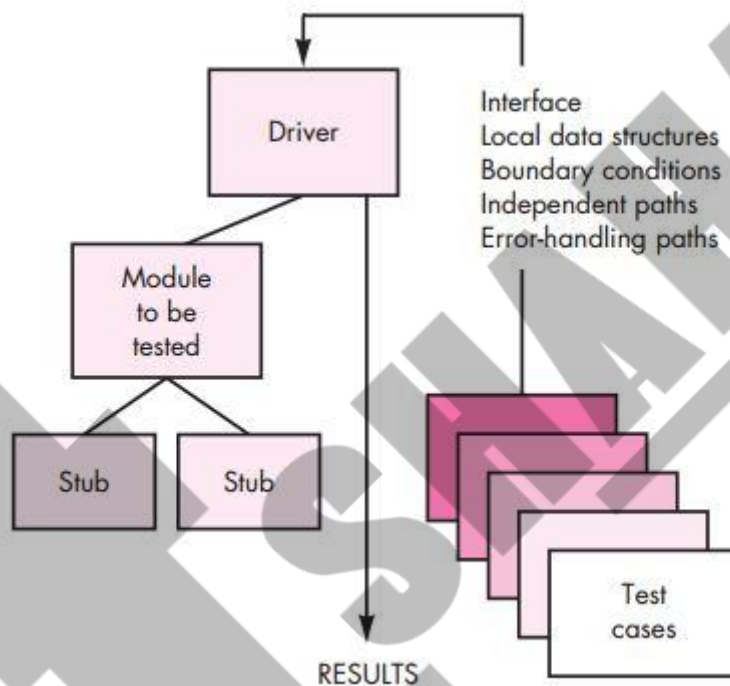
Because a component is not a stand-alone program, driver and/or stub software must often be developed for each unit test. In most applications a driver is nothing more than a “main program” that accepts test case data, passes such data to the component (to be tested), and prints relevant results. Stubs serve to replace

Solved by Talha Shahab

modules that are subordinate (invoked by) the component to be tested. A stub or “dummy subprogram” uses the subordinate module’s interface, may do minimal data manipulation, prints verification of entry, and returns control to the module undergoing testing.

Drivers and stubs represent testing “overhead.” That is, both are software that must be written (formal design is not commonly applied) but that is not delivered with the final software product. If drivers and stubs are kept simple, actual overhead is relatively low. Unfortunately, many components cannot be adequately unit tested with “simple” overhead software. In such cases, complete testing can be postponed until the integration test step (where drivers or stubs are also used).

Unit testing is simplified when a component with high cohesion is designed. When only one function is addressed by a component, the number of test cases is reduced and errors can be more easily predicted and uncovered.



Integration Testing

Integration testing is a systematic technique for constructing the software architecture while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit-tested components and build a program structure that has been dictated by design.

There is often a tendency to attempt non-incremental integration; that is, to construct the program using a “big bang” approach. All components are combined in advance. The entire program is tested as a whole. And chaos usually results! A set of errors is encountered. Correction is difficult because isolation of causes is complicated by the vast expanse of the entire program. Once these errors are corrected, new ones appear and the process continues in a seemingly endless loop.

Incremental integration is the antithesis of the big bang approach. The program is constructed and tested in small increments, where errors are easier to isolate and correct; interfaces are more likely to be tested completely; and a systematic test approach may be applied. In the paragraphs that follow, a number of different incremental integration strategies are discussed.

➤ Top-down integration

Top-down integration testing is an incremental approach to construction of the software architecture.

Modules are integrated by moving downward through the control hierarchy, beginning with the main control

Solved by Talha Shahab

module (main program). Modules subordinate (and ultimately subordinate) to the main control module are incorporated into the structure in either a depth-first or breadth-first manner. The integration process is performed in a series of five steps:

- (1) The main control module is used as a test driver and stubs are substituted for all components directly subordinate to the main control module.
- (2) Depending on the integration approach selected (i.e., depth or breadth first), subordinate stubs are replaced one at a time with actual components.
- (3) Tests are conducted as each component is integrated.
- (4) On completion of each set of tests, another stub is replaced with the real component.
- (5) Regression testing (discussed later in this section) may be conducted to ensure that new errors have not been introduced.

The process continues from step 2 until the entire program structure is built. The top-down integration strategy verifies major control or decision points early in the test process. In a “well-factored” program structure, decision making occurs at upper levels in the hierarchy and is therefore encountered first. If major control problems do exist, early recognition is essential. If depth-first integration is selected, a complete function of the software may be implemented and demonstrated. Early demonstration of functional capability is a confidence builder for all stakeholders.

Top-down strategy sounds relatively uncomplicated, but in practice, logistical problems can arise. The most common of these problems occurs when processing at low levels in the hierarchy is required to adequately test upper levels. Stubs replace low-level modules at the beginning of top-down testing; therefore, no significant data can flow upward in the program structure. As a tester, you are left with three choices:

- (1) Delay many tests until stubs are replaced with actual modules,
- (2) Develop stubs that perform limited functions that simulate the actual module,
- (3) Integrate the software from the bottom of the hierarchy upward.

The first approach (delay tests until stubs are replaced by actual modules) can cause you to lose some control over correspondence between specific tests and incorporation of specific modules. The second approach is workable but can lead to significant overhead, as stubs become more and more complex. The third approach, called bottom-up integration is discussed in the paragraphs that follow.

➤ **Bottom-up integration**

Bottom-up integration testing, as its name implies, begins construction and testing with atomic modules (i.e., components at the lowest levels in the program structure). Because components are integrated from the bottom up, the functionality provided by components subordinate to a given level is always available and the need for stubs is eliminated. A bottom-up integration strategy may be implemented with the following steps:

- (1) Low-level components are combined into clusters (sometimes called builds) that perform a specific software sub-function.
- (2) A driver (a control program for testing) is written to coordinate test case input and output.
- (3) The cluster is tested.
- (4) Drivers are removed and clusters are combined moving upward in the program structure.

Regression testing

Each time a new module is added as part of integration testing, the software changes. New data flow paths are established, new I/O may occur, and new control logic is invoked. These changes may cause problems with functions that previously worked flawlessly. In the context of an integration test strategy, regression testing is the re-execution of some subset of tests that have already been conducted to ensure that changes have not propagated unintended side effects.

In a broader context, successful tests (of any kind) result in the discovery of errors, and errors must be corrected. Whenever software is corrected, some aspect of the software configuration (the program, its documentation, or the data that support it) is changed. Regression testing helps to ensure that changes (due to testing or for other reasons) do not introduce unintended behavior or additional errors.

- ✓ A representative sample of tests that will exercise all software functions.
- ✓ Additional tests that focus on software functions that are likely to be affected by the change.
- ✓ Tests that focus on the software components that have been changed.

As integration testing proceeds, the number of regression tests can grow quite large. Therefore, the regression test suite should be designed to include only those tests that address one or more classes of errors in each of the major program functions. It is impractical and inefficient to re-execute every test for every program function once a change has occurred.

Smoke testing

Smoke testing is an integration testing approach that is commonly used when product software is developed. It is designed as a pacing mechanism for time-critical projects, allowing the software team to assess the project on a frequent basis. In essence, the smoke-testing approach encompasses the following activities:

- (1) Software components that have been translated into code are integrated into a build. A build includes all data files, libraries, reusable modules, and engineered components that are required to implement one or more product functions.
- (2) A series of tests is designed to expose errors that will keep the build from properly performing its function. The intent should be to uncover “showstopper” errors that have the highest likelihood of throwing the software project behind schedule.
- (3) The build is integrated with other builds, and the entire product (in its current form) is smoke tested daily. The integration approach may be top down or bottom up.

The daily frequency of testing the entire product may surprise some readers. However, frequent tests give both managers and practitioners a realistic assessment of integration testing progress. Smoke testing provides a number of benefits when it is applied on complex, time critical software projects:

- ✓ Integration risk is minimized. Because smoke tests are conducted daily, incompatibilities and other show-stopper errors are uncovered early, thereby reducing the likelihood of serious schedule impact when errors are uncovered.
- ✓ The quality of the end product is improved. Because the approach is construction (integration) oriented, smoke testing is likely to uncover functional errors as well as architectural and component-level design errors. If these errors are corrected early, better product quality will result.
- ✓ Error diagnosis and correction are simplified. Like all integration testing approaches, errors uncovered during smoke testing are likely to be associated with “new software increments”—that is, the software that has just been added to the build(s) is a probable cause of a newly discovered error.
- ✓ Progress is easier to assess. With each passing day, more of the software has been integrated and more has been demonstrated to work. This improves team morale and gives managers a good indication that progress is being made.

Q6. What is Project Management? Explain Project Management Life Cycle?

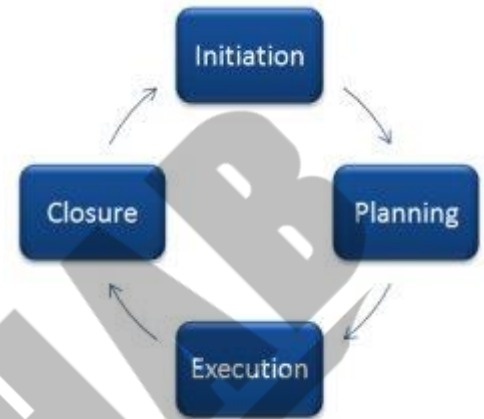
Solved by Talha Shahab

Project management is the practice of initiating, planning, executing, controlling, and closing the work of a team to achieve specific goals and meet specific success criteria at the specified time. A project is a temporary endeavor designed to produce a unique product, service or result with a defined beginning and end (usually time-constrained, and often constrained by funding or staffing) undertaken to meet unique goals and objectives, typically to bring about beneficial change or added value.

The Project Life Cycle

The project management life cycle provides a framework for managing any type of project. By definition, a project has a definite beginning and end. Between the beginning and end points, the project can be divided into four phases...

- Project Initiation
- Project Planning
- Project Execution
- Project Closure



Project Initiation

The purpose of the Project Initiation Phase is to define and authorize the project. The initial definition of the project can come from several places...

- ✓ Project Statement of Work (SoW)
- ✓ Business Case
- ✓ Contract

The project manager takes the information provided and creates a Project Charter. The Project Charter authorizes the project and documents the initial requirements for the project. It generally includes information such as...

- ✓ Project purpose, vision, and mission
- ✓ Measurable objectives and success criteria
- ✓ High level project description, requirements, and risks
- ✓ Summary milestone schedule and budget
- ✓ Name and authority of the project sponsor

An important part of starting your project off right is performing a stakeholder analysis. Understanding which people or organizations will be impacted by or can influence your project is critical for ensuring your project's success.

Project Planning

The purpose of the Project Planning Phase is to determine the approach you will take and define all the details of how the project will be done. Project Planning has two parts...

- ✓ Strategic Planning
- ✓ Implementation Planning.

During Strategic Planning you develop the overall approach to the project. During Implementation Planning you figure out all the details of how the project will be done. A good way to visualize this is to think of your project as a family vacation.

- ✓ During Project Initiation you determine where you want to go (your mission).
- ✓ During Strategic Planning, you decide whether you want to fly there or drive (your approach).

Let's say you decide to drive. In that case, during Implementation Planning you would map out your route, identify which hotels you will stay at along the way, determine how long each leg of the trip will take, and so on (all the details).

Project Execution

The purpose of the Project Execution Phase is to carry out the activities defined during the Project Planning Phase. Project Execution is where most of the time, money, and people are used on a project. This is where the action takes place.

During this phase of the project management life cycle the project manager has to keep all the activities moving forward in a coordinated manner. This means you will need to track the progress of each activity and adjust your plans when the situation changes. This tracking and adjustment of project activities is also known as Monitor and Control.

During the execution phase all of the agreed project deliverables should be implemented and accepted by the customer. The customer can be an internal customer or an external customer.

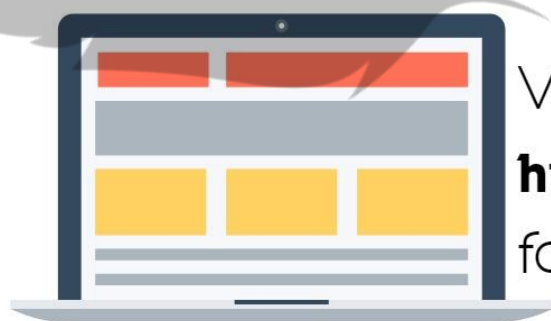
Project Closure

The purpose of the Project Closure Phase is to formally close the project. During Project Closure, there are several key activities that need to be performed...

- ✓ Verify that the completion criteria are met
- ✓ Create a project closure report
- ✓ Collect and archive project artifacts
- ✓ Perform a project postmortem

Many projects skip this phase. Once the Execution Phase is complete, they simply move on. It's unfortunate since they really don't know if the project objectives have been met, don't organize the project artifacts to be easily found for future project's reference, and don't identify the key issues and lessons learned by the project that can be applied to future projects.

Performing Project Closure will benefit both your company and your career. If you do this well, you will set yourself up to lead high-visibility, business-critical projects. So make sure your projects go through the full project management life cycle.



Visit

<https://tshahab.blogspot.com>

for more.