✆ STD - 0431 - 2670142
FAX - 0431 - 2670143

DHANALAKSHMI SRINIVASAN
INSTITUTE OF TECHNOLOGY
(Approved by AICTE, New Delhi & Affiliated to Anna University)
NH - 45, Trichy - Chennai Trunk Road,
SAMAYAPURAM,TRICHY - 621 112.
E.mail:dsit2011@gmail.com Website:www.dsit.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
QUESTION WITH ANSWER
CS8404-SOFTWARE ENGINEERING

UNIT I

1. Write the IEEE definition of software engineering

Software Engineering: (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1).

2. Mention the characteristics of software contrasting it with characteristics of hardware

- Software is developed or engineered; it is not manufactured in the classical sense.
- Software doesn't -wear out
- Although the industry is moving toward component-based construction, most software continues to be custom built

3. 'Software doesn't wear out' justify.

Software doesn't wear out, but it does deteriorate. During the software life, it will undergo change (maintenance). As changes are made, it is likely that some new defects will be introduced, causing the failure rate spike

4. Define process

- A process is a collection of activities, actions, and tasks that are performed when Some work product is to be created.
- An activity strives to achieve a broad objective (e.g., communication with stakeholders) An action encompasses a set of tasks that produce a major work product (e.g., an architectural design model).
- A task focuses on a small, but well-defined objective (e.g., conducting a unit test) that produces a tangible outcome.

5. List out the activities of generic process framework

- Communication,
- Planning,
- Modeling,
- Construction
- Deployment

6. List out the umbrella activities of software engineering process

- Software project tracking and control
- Risk management

- Software quality assurance
- Technical reviews
- Measurement
- Software configuration management
- Reusability management
- Work product preparation and production

7. What are the steps involved in component based development model

- Available component-based products are researched and evaluated for the
- Application domain in question.
- Component integration issues are considered.
- Software architecture is designed to accommodate the components.
- Components are integrated into the architecture.
- Comprehensive testing is conducted to ensure proper functionality.

8. What is process pattern?

A process pattern describes the process related problem that encountered during software engineering work, identifies the environment in which the problem has been encountered and suggests one or more proven solutions to the problem

9. What are the different types of process pattern?

- *Stage pattern*—defines a problem associated with a framework activity for the process. An example of a stage pattern might be **EstablishingCommunication.**
- *Task pattern*—defines a problem associated with a software engineering action or work task and relevant to successful software engineering practice (e.g.**RequirementsGathering** is a task pattern).
- *Phase pattern*—define the sequence of framework activities that occurs within the process, even when the overall flow of activities is iterative in nature. An example of a phase pattern might be **SpiralModel** or **Prototyping**

10.Mention the drawbacks of formal method.

- The development of formal models is currently quite time consuming and expensive.
- Because few software developers have the necessary background to apply formal methods, extensive training is required.
- It is difficult to use the models as a communication mechanism for technically unsophisticated customers.

11.What is prototyping?

A customer defines a set of general objectives for software, but does not identify detailed requirements for functions and features. In other cases, the developer may be unsure of the efficiency of an algorithm, the adaptability of an operating system, or the form that human- machine interaction should take. In these, and many other

situations, a *prototyping paradigm* may offer the best approach. Although prototyping can be used as a stand-alone process model, it is more commonly used as a technique that can be implemented within the context of any one of the process models. **The prototyping paradigm assists you and other stakeholders to better understand what is to be built when requirements are fuzzy**

12.Mention the drawbacks of water fall model

a. Real projects rarely follow the sequential flow that the model proposes.
b. It is often difficult for the customer to state all requirements explicitly.
c. The customer must have patience. A working version of the program(s) will not be available until late in the project time span.
d. There is no back tracking

13.What are the advantages of incremental model?

- The *incremental* model combines elements of linear and parallel process flows.
- Incremental development is particularly useful when staffing is unavailable for a complete implementation by the business deadline that has been established for the project.
- Core product is delivered to the user in the first increment. So that changes any can be made in the successive increments
- Increments can be planned to manage technical risks..

14.List the "Manifesto for Agile Software Development

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

15.What is an Agile Process?

- Is driven by customer descriptions of what is required (scenarios)
- Recognizes that plans are short-lived
- Develops software iteratively with a heavy emphasis on construction activities
- Delivers multiple _software increments' □
- Adapts as changes occur

16.What is an Agility in context of software engineering

- Agility means effective (rapid and adaptive) response to change, effective communication among all stockholder.
- Drawing the customer onto team and organizing a team so that it is in control of work performed. -The Agile process, light-weight methods are People-based rather than plan-based methods.
- The agile process forces the development team to focus on software itself rather than design and documentation.
- The agile process believes in iterative method.

- The aim of agile process is to deliver the working model of software quickly to the customer For example: Extreme programming is the best known of agile process.

17.List out the XP values

Communication, Simplicity, Feedback, Courage, Respect

UNIT II

1. **What is functional requirement?**

These are the Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations. May state what the system should not do.

2. **What is non functional requirement?**

These are Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc. Often apply to the system as a whole rather than individual features or services

3. **What are the types of requirements?**
   - User requirements

     Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers.
   - System requirements

     A structured document setting out detailed descriptions of the system's functions, services and operational constraints. Defines what should be implemented so may be part of a contract between client and contractor.
4. **How will you classify non functional requirements?**
   - Product requirements - Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.
   - Organizational requirements- Requirements which are a consequence of organisational policies and procedures e.g. process standards used, implementation requirements, etc
   - External requirements - Requirements which arise from factors which are external to the system and its development process e.g. interoperability requirements, legislative requirements, etc.
5. **What are the properties of non functional requirements?**
   - Speed
   - Size
   - Ease of use
   - Reliability
   - Robustness
   - Portability
6. **What is SRS?**

The software requirements document is the official statement of what is required of the system developers. It should include both a definition of user

requirements and a specification of the system requirements. It is NOT a design document. As far as possible, it should set of WHAT the system should do rather than HOW it should do it.

7. **What are the activities included in requirement engineering process**
   - Business (feasibility study)
   - Discovering requirements (elicitation and analysis)
   - Converting these requirements into some standard form (specification)
   - Checking that the requirements actually define the system that the customer wants (validation**)**
8. **What are the problems of requirement analysis or elicitation?**
   - Stakeholders don't know what they really want.
   - Stakeholders express requirements in their own terms.
   - Different stakeholders may have conflicting requirements.
   - Organisational and political factors may influence the system requirements.
   - The requirements change during the analysis process. New stakeholders may emerge and the business environment may change.
9. **What is feasibility study?**

A feasibility study is a short, focused study that should take place early in the requirement engineering process. It should answer three key questions.

- Does the system contribute to the overall objective of the organization?
- Can the system be implemented within schedule and budget using current technology?
- Can the system be integrated with other systems that are used?

10. **What do you mean by stakeholders?**

Stakeholder is anyone who must have some direct or indirect influence on the system requirements. Stakeholder includes end users who will interact with the system and anyone else in an organization who will be affected by it. Other system stakeholder's might be engineers who are developing or maintaining other related systems, business managers, domain experts and trade union representative.

11. **What is requirement elicitation or discovery?**

The process of gathering information about the required and existing systems and distilling the user and system requirements from this information. Source of the information during the requirements discovery phase include documentation, system stake holders and specification of similar systems.

12. **What are the types of interviews?**
    - Closed interviews based on pre-determined list of questions
    - Open interviews where various issues are explored with stakeholders.
13. **What is scenario?**

- Scenarios are real-life examples of how a system can be used.
- They should include
  - A description of the starting situation;
  - A description of the normal flow of events;
  - A description of what can go wrong;

- Information about other concurrent activities;
- A description of the state when the scenario finishes.

14. What is use case?

Use-cases are a scenario based technique in the UML which identify the actors in an interaction and which describe the interaction itself. A set of use cases should describe all possible interactions with the system. High-level graphical model supplemented by more detailed tabular description. Sequence diagrams may be used to add detail to use-cases by showing the sequence of event processing in the system.

15. How the requirements are validated?

It is the process of checking that requirements actually define the system that the customer really wants. It overlaps with analysis as it is concerned with finding problems with the requirements.

Validity check

Consistency check
Completeness checks
Realism checks
Verifiability

16. List out the requirement validation techniques

- Requirements reviews- Systematic manual analysis of the requirements.
- Prototyping- Using an executable model of the system to check requirements.
  Test-case generation - Developing tests for requirements to check testability.

17. What is data dictionary?

A data dictionary stores information about data items found in a DFD

- Name – identifies data item
- Alias – identifies other names
- Data structure (type) – type of data(integer, char)
- Description – indicates how a data item is used
- Duration – life span of data
- Accuracy- high, medium low
- Range of values – allowable values of data item
- Data flow – identifies process that generate data

18. What is a Petri net?

A petri net is a form of finite state machine useful in modeling concurrency and asynchronous communication. It is used to describe and analyze the structure and information flow in the system

19. "An SRS is traceable ". Comment.

An SRS is traceable if the origin of each of its requirement is clear. Traceability is of two types. They are **a. Forward traceability b. backward traceability**

Forward traceability means that each requirement should be traceable to some design and code elements. Backward traceability requires that it be possible to trace design and code elements to the requirement they support

UNIT III

1. **What are the golden rules of an interface design?**
   - Place the user in control.
   - Reduce the user's memory load.
   - Make the interface consistent.

2. **Write note on FURPS model of design quality.**
   - *Functionality* is assessed by evaluating the feature set and capabilities of the program, the generality of the functions that are delivered, and the security of the overall system.
   - *Usability* is assessed by considering human factors , overall aesthetics, consistency, and documentation.
   - *Reliability* is evaluated by measuring the frequency and severity of failure, the accuracy of output results, the mean-time-to-failure (MTTF), the ability to recover from failure, and the predictability of the program.
   - *Performance* is measured by considering processing speed, response time, Resource consumption, throughput, and efficiency.
   - *Supportability* combines the ability to extend the program (extensibility), adaptability, serviceability

3. **What is cohesion?**

Cohesion as the "single-mindedness" of a component. Within the context of component-level design for object-oriented systems, *cohesion* implies that a component or class encapsulates only attributes and operations that are closely related to one another and to the class or component itself. **Three types of cohesion are**

Functional, layer and communicational

4. **What is coupling?**

Coupling is a qualitative measure of the degree to which classes are connected to one another. As classes (and components) become more interdependent, coupling increases. An important objective in component-level design is to keep coupling as low as is possible.

Types of coupling are

Content, common, control, stamp, data, routine call, type use, inclusion and external coupling

5. **What is design process?**

Software design is an iterative process through which requirements are translated into a "blueprint" for constructing the software. Initially, the blueprint depicts a holistic view of software. That is, the design is represented at a high level of abstraction— a level that can be directly traced to the specific system objective and more detailed data, functional, and behavioral requirements. As design iterations occur, subsequent refinement leads to design representations at much lower levels of abstraction. These can still be traced to requirements, but the connection is more subtle.

6. Define abstraction with its types

At the highest level of abstraction, a solution is stated in broad terms using the language of the problem environment. At lower levels of abstraction, a more detailed description of the solution is provided.

A *procedural abstraction* refers to a sequence of instructions that have a specific and limited function. Eg. *Open* for a door.

A *data abstraction* is a named collection of data that describes a data object. In the context of the procedural abstraction *open,* we can define a data abstraction called **door.**

7. What is software architecture?

The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them.

8. Define modularity.

Modularity is the most common manifestation of separation of concerns. Software is divided into separately named and addressable components, sometimes called *modules that* are integrated to satisfy problem requirements. It has been stated that "modularity is the single attribute of software that allows a program to be intellectually manageable"

9. What is refactoring?

"Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code [design] yet improves its internal structure."

10. What is archetypes and list out its different types?

An *archetype* is a class or pattern that represents a core abstraction that is critical to the design of architecture for the target system. In general, a relatively small set of archetypes is required to design even relatively complex systems.

Node, detector, indicator and controller

11. List out the generic structure of architectural context diagram

- *Super ordinate systems*—those systems that use the target system as part of some higher-level processing scheme.
- *Subordinate systems*—those systems that are used by the target system and provide data or processing that are necessary to complete target system functionality.
- *Peer-level systems*—those systems that interact on a peer-to-peer basis (i.e., information is either produced or consumed by the peers and the target system.
- *Actors*—entities (people, devices) that interact with the target system by producing or consuming information that is necessary for requisite processing.

12. What are the elements of interface design?

1) the user interface (UI);
2) External interfaces to other systems, devices, networks, or other producers or consumers of information;
(3) Internal interfaces between various design components.

13. List any four technical criteria for good design

- A design should be modular; that is, the software should be logically partitioned into elements or subsystems.
- A design should contain distinct representations of data, architecture, interfaces, and components.
- A design should lead to components that exhibit independent functional characteristics.
- A design should lead to interfaces that reduce the complexity of connections between components and with the external environment.

14. How pattern differs from a style?

A pattern differs from a style in a number of fundamental ways:
(1) The scope of a pattern is less broad, focusing on one aspect of the architecture rather than the architecture in its entirety;
(2) A pattern imposes a rule on the architecture, describing how the software will handle some aspect of its functionality at the infrastructure level (e.g., concurrency)
(3) Architectural patterns tend to address specific behavioral issues within the context of the architecture. Patterns can be used in conjunction with an architectural style to shape the overall structure of a system.

15. What is the need for architectural mapping using dataflow?

A mapping technique, called *structured design* is often characterized as a data flow- oriented design method because it provides a convenient transition from a data flow diagram to software architecture. The transition from information flow (represented as a DFD) to program structure is accomplished as part of a six step process: (1) the type of information flow is established, (2) flow boundaries are indicated, (3) the DFD is mapped into the program structure, (4) control hierarchy is defined, (5) the resultant structure is refined using design measures and heuristics, and (6) the architectural description is refined and elaborated.

16. What is open closed principle?

"A module [component] should be open for extension but closed for modification". This statement seems to be a contradiction, but it represents one of the most important characteristics of a good component-level design. Stated simply, you should specify the component in a way that allows it to be extended (within the functional domain that it addresses) without the need to make internal (code or logic- level) modifications to the component itself. To accomplish this, you create abstractions that serve as a buffer between the functionality that is likely to be extended and the design class itself.

17. List out the steps applied to develop a decision table?

- List all actions that can be associated with a specific procedure (or

component).

- List all conditions (or decisions made) during execution of the procedure.
- Associate specific sets of conditions with specific actions, eliminating impossible combinations of conditions; alternatively, develop every possible permutation of conditions.
- Define rules by indicating what actions occur for a set of conditions.

18. List various types of architectural styles

- Data centered architecture
- Data flow architecture
- Call and return architecture
- Object oriented architecture
- Layered architecture

UNIT IV

1. List out the characteristics of testable software

Operability, Observability, Controllability, Decomposability, Simplicity, Stability, Understandability

2. Describe the objective of testing.

a. Testing is a process of executing a program with the intent of finding an error.
b. A good test case is one that has a high probability of finding an as-yet undiscovered error.
c. A successful test is one that uncovers an as-yet-undiscovered error

3. List the attributes of a good test

- A good test has a high probability of finding an error.
- A good test is not redundant.
- A good test should be “best of breed”
- A good test should be neither too simple nor too complex.

4. Differentiate verification and validation

Verification refers to the set of activities that ensure that software correctly implements a specific function.

- Verification: "Are we building the product right?"

Validation refers to a different set of activities that ensure that the software that has been built is traceable to customer requirements.

- Validation: "Are we building the right product?“

5. What is white box testing? Or what is internal view of testing?

Knowing the internal workings of a product, tests can be conducted to ensure that “all gears mesh,” that is, internal operations are performed according to specifications and all internal components have been adequately exercised. This approach takes an internal view and is termed white-box testing.

6. What is black box testing? Or what is external view of testing?

Knowing the specified function that a product has been designed to perform, tests can be conducted that demonstrate each function is fully operational while at the same time searching for errors in each function. This approach takes an external view and is called black-box testing.

7. Mention the white box testing methods.

Using white-box testing methods, you can derive test cases that

(1) Guarantee that all independent paths within a module have been exercised at least once,
(2) Exercise all logical decisions on their true and false sides,
(3) Execute all loops at their boundaries and within their operational bounds, and
(4) Exercise internal data structures to ensure their validity.

8. What is basis path testing and list its methods

Basis path testing is a white-box testing technique. The basis path method enables the test-case designer to derive a logical complexity measure of a procedural design and use this measure as a guide for defining a basis set of execution paths. Test cases derived to exercise the basis set are guaranteed to execute every statement in the program at least one time during testing.

- Flow graph notation
- Independent program paths
- Deriving test cases
- Graph matrix

9. What is cyclomatic complexity?

Cyclomatic complexity is software metric that provides a quantitative measure of the logical complexity of a program. When used in the context of the basis path testing method, the value computed for cyclomatic complexity defines the number of independent paths in the basis set of a program and provides you with an upper bound for the number of tests that must be conducted to ensure that all statements have been executed at least once.

10. How will you compute cyclomatic complexity?

Cyclomatic complexity has a foundation in graph theory and provides you with extremely useful software metric. Complexity is computed in one of three ways:

1. The number of regions of the flow graph corresponds to the cyclomatic complexity.
2. Cyclomatic complexity $V(G)$ for a flow graph G is defined as

$$\boldsymbol{V(G) = E - N + 2}$$

Where E is the number of flow graph edges and N is the number of flow graph nodes.
3. Cyclomatic complexity $V(G)$ for a flow graph G is also defined as

$$\boldsymbol{V(G) = P + 1}$$

where P is the number of predicate nodes contained in the flow graph G.

11. What is conditional testing?

Condition testing is a test-case design method that exercises the logical conditions contained in a program module. A simple condition is a Boolean variable or a relational expression, possibly preceded with one NOT (¬) operator. A relational expression takes the form

***E*1 <relational-operator> *E*2**

where *E*1 and *E*2 are arithmetic expressions and <relational-operator> is one of the following: <, <=, =, >, >= , not equal .

12. What is a loop testing and list different type of loop?

Loop testing is a white-box testing technique that focuses exclusively on the validity of loop constructs. Four different classes of loops can be defined: Simple loops, concatenated loops, nested loops, and unstructured loops

13. What types of errors will be found by black box testing?

a. Incorrect or missing functions,
b. Interface errors,
c. Errors in data structures or external data base access.
d. Behavior or performance errors,
e. Initialization and termination errors.

14.* What is Equivalence partitioning*?

Equivalence partitioning is a black-box testing method that divides the input domain of a program into classes of data from which test cases can be derived. Test case design for equivalence partitioning is based on an evaluation of equivalence classes for an input condition. An equivalence class represents a set of valid or invalid states for input conditions. Typically, an input condition is a specific numeric value, a range of values, a set of related values, or a Boolean condition.

15. What is regression testing?

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. Regression testing may be conducted manually, by re-executing a subset of all test cases or using automated capture/playback tools. The Capture/playback tools enable the software engineer to capture test cases and results for subsequent playback and comparison.

16. What is alpha testing?

The *alpha test* is conducted at the developer's site by a customer. The software is used in a natural setting with the developer "looking over the shoulder" of the user and recording errors and usage problems. Alpha tests are conducted in a controlled environment.

17. What is beta testing?

The *beta test* is conducted at one or more customer sites by the end-user of the

software. Beta test is a "live" application of the software in an environment that cannot be controlled by the developer. The customer records all problems (real or imagined) that are encountered during beta testing and reports these to the developer at regular intervals. As a result of problems reported during beta tests, software engineers make modifications and then prepare for release of the software product to the entire customer base.

18. List out various types of system testing

Types of system tests are:

a. Recovery Testing
b. Security Testing
c. Stress Testing\
d. Performance Testing

19. Why debugging is difficult?

- The symptom may disappear (temporarily) when another error is corrected.
- The symptom may actually be caused by non-errors (e.g., round-off inaccuracies).
- The symptom may be caused by human error that is not easily traced (e.g. wrong input, wrongly configure the system)
- The symptom may be a result of timing problems, rather than processing problems.(

e.g. taking so much time to display result).

- It may be difficult to accurately reproduce input conditions (e.g., a real-time application in which input ordering is indeterminate).

20. What is debugging and list out its strategies

Debugging is the process that results in the removal of the error. Although debugging can and should be an orderly process, it is still very much an art. Debugging is not testing but always occurs as a consequence of testing.

Three categories for debugging approaches o Brute force

- o Backtracking
- Cause elimination

21. List out the activities of BPmodel

- Business definition
  - Process identification
  - Process evaluation
  - Process specification and design
  - Prototyping
  - Refinement and instantiation

UNIT V

1. **Compare size oriented and function oriented metrics.**

   Size oriented metrics attempt to quantify software project by using the size of the project to normalize other quality measures whereas function oriented metrics attempt to measure the functionality of a software system.

2. **An Organic software occupies 15,000 LOC. How many programmers are needed to complete?**

Programmers needed = 2.4 * (15)^1.05

3. **What are the different types of productivity estimates?**

   Productivity estimates are usually based on measuring attributes of the software and dividing this by the total effort required for development. Software metrics have two main types:

- Size-related software metrics. These metrics are related to the size of software. The most frequently used size-related metric is lines of delivered source code.
- Function-related software metrics. These are related to the overall functionality of the software. For example, function points and object points are metrics of this type.

4. **What is the difference between project and process metrics?**

   Process metrics are collected across all projects and over long periods of time. Their intent is to provide a set of process indicators that lead to long-term software process improvement. Project metrics enable a software project manager to (1) assess the status of an ongoing project, (2) track potential risks, (3) uncover problem areas before they go

-critical,|| (4) adjust work flow or tasks, and (5) evaluate the project team's ability to control quality of software work products.

5. **What is task set?**

   A task set is a collection of software engineering work tasks, milestones, work products, and quality assurance filters that must be accomplished to complete a particular project. The task set must provide enough discipline to achieve high software quality. But, at the same time, it must not burden the project team with unnecessary work.

6. **What is task network?**

   A task network, also called an activity network, is a graphic representation of the task flow for a project. It is sometimes used as the mechanism through which task sequence and dependencies are input to an automated project scheduling tool. In its simplest form (used when creating a macroscopic schedule), the task network depicts major software engineering actions.

7. **List out the steps in project planning**

- Establish project scope
- Determine feasibility
- Analyze risks

- Define required resources
    1. Determine require human resources
    2. Define reusable software resources
    3. Identify environmental resources
- Estimate cost and effort
- Develop a project schedule

8. What is earned value and earned value analysis?

The earned value system provides a common value scale for every task, regardless of the type of the work being performed. The total hours to do the whole project are estimated and every task is given earned values based on its estimated percentage of the total

A technique for performing quantitative analysis of the progress is called earned value analysis

9. How the effort are distributed in the scheduling.

A recommended distribution of effort across the software process is often referred to as the *40–20–40 rule.* Forty percent of all effort is allocated to frontend analysis and design. A similar percentage is applied to back-end testing. You can correctly infer that coding (20 percent of effort) is deemphasized.

10. What is project scheduling?

Software project scheduling is an action that distributes estimated effort across the planned project duration by allocating the effort to specific software engineering tasks.

11. Define risk

- Risk concerns future happenings.
- The risk involves change, such as in changes of mind, opinion, actions, or places.
- Risk involves choice, and the uncertainty that choice itself entails.

12. What are the characteristics of risk?

- *uncertainty*—the risk may or may not happen; that is, there are no 100 percent probable risks1—and
- *loss*—if the risk becomes a reality, unwanted consequences or losses will occur

13. What are the different types of risk?

Project risks Technical risks
Business risks Known risk
Predictable risk
Unpredictable risk

14. What are the steps involved in risk projection?

- Establish a scale that reflects the perceived likelihood of a risk.
- Delineate the consequences of the risk.
- Estimate the impact of the risk on the project and the product.
- Assess the overall accuracy of the risk projection so that there will be no misunderstandings.

15. How will measure risk exposure (RE)?

The overall *risk exposure* RE is determined using the following relationship
RE = *P* * *C* where *P* is the probability of occurrence for a risk, and *C* is the cost to the project should the risk occur