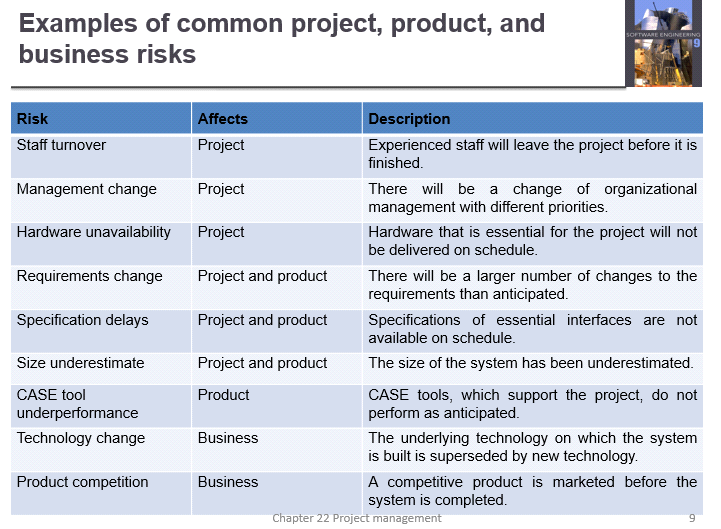
**1) Explain different types of S/W risks?**

**2) Explain risk management in detail?**

Answer:

* Risk management is concerned with identifying risks and drawing up plans to minimise their effect on a project.
* A risk is a probability that some adverse circumstance will occur
* Project risks affect schedule or resources;
* Product risks affect the quality or performance of the software being developed;
* Business risks affect the organisation developing or procuring the software.



**The risk management process**

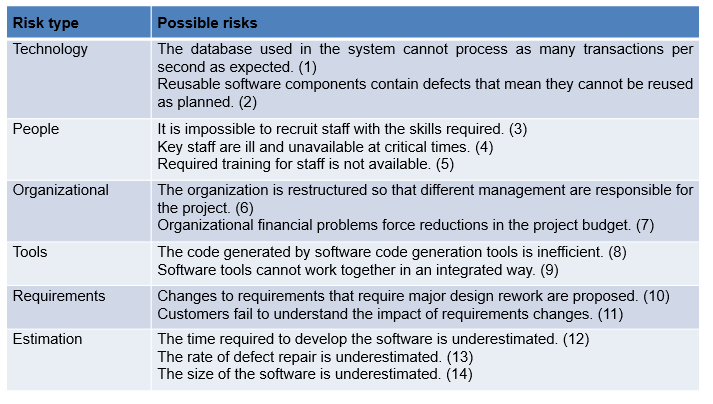
* Risk identification
* Identify project, product and business risks;
* Risk analysis
* Assess the likelihood and consequences of these risks;
* Risk planning
* Draw up plans to avoid or minimise the effects of the risk;
* Risk monitoring
* Monitor the risks throughout the project;



**Risk identification**

* May be a team activities or based on the individual project manager’s experience.
* A checklist of common risks may be used to identify risks in a project
* Technology risks.
* People risks.
* Organisational risks.
* Requirements risks.
* Estimation risks.

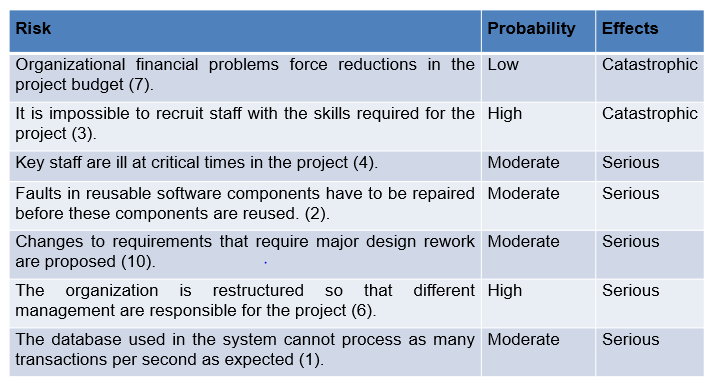
**Examples of different risk types**

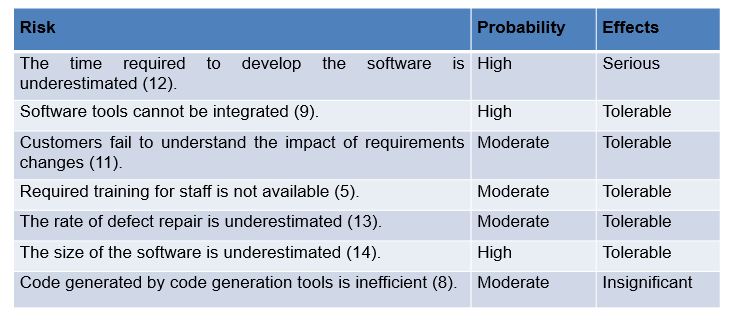


**Risk analysis**

* Assess probability and seriousness of each risk.
* Probability may be very low, low, moderate, high or very high.
* Risk consequences might be catastrophic, serious, tolerable or insignificant.

**Risk types and examples**

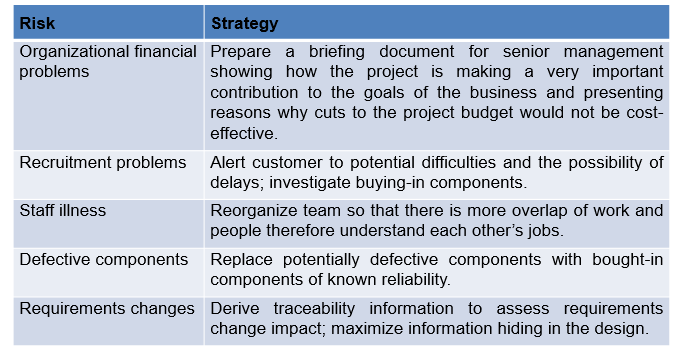


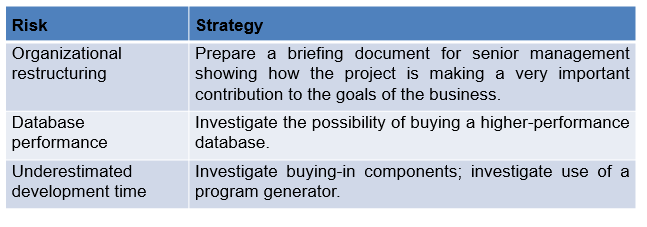


**Risk planning**

* Consider each risk and develop a strategy to manage that risk.
* Avoidance strategies
* The probability that the risk will arise is reduced;
* Minimisation strategies
* The impact of the risk on the project or product will be reduced;
* Contingency plans
* If the risk arises, contingency plans are plans to deal with that risk;

**Strategies to help manage risk**

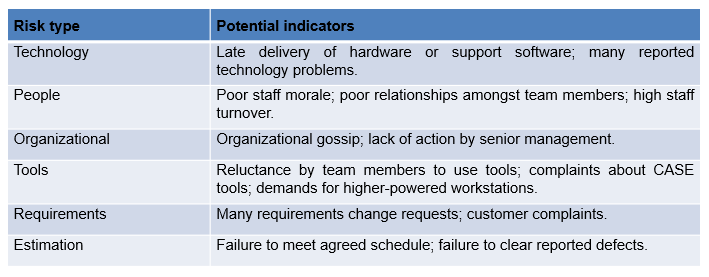




**Risk monitoring**

* Assess each identified risks regularly to decide whether or not it is becoming less or more probable.
* Also assess whether the effects of the risk have changed.
* Each key risk should be discussed at management progress meetings.

**Risk indicators**



* Good project management is essential if software engineering projects are to be developed on schedule and within budget.
* Software management is distinct from other engineering management. Software is intangible. Projects may be novel or innovative with no body of experience to guide their management. Software processes are not as mature as traditional engineering processes.
* Risk management is now recognized as one of the most important project management tasks.
* Risk management involves identifying and assessing project risks to establish the probability that they will occur and the consequences for the project if that risk does arise. You should make plans to avoid, manage or deal with likely risks if or when they arise.

**3) Explain about requirement engineering process?**

4.6 RE-process.eps                                             000FF8ECMacintosh HD                   B8AA5F2E:

* Feasibility-the study considers whether the proposed system will be cost effective from a business point of view and whether it can be developed within existing budgetary constraints.
* Elication-this is the process of deriving the system requirements through observation of existing systems.

**4) Explain what are various S/W process model with a neat sketch?**

* A structured set of activities required to develop a   
  software system
  + Specification;
  + Design;
  + Validation;
  + Evolution.
* A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective.

**Generic software process models**

* The waterfall model
  + Separate and distinct phases of specification and development.
* Evolutionary development
  + Specification, development and validation are interleaved.
* Component-based software engineering
  + The system is assembled from existing components.

There are many variants of these models e.g. formal development where a waterfall-like process is used but the specification is a formal specification that is refined through several stages to an implementable design



* Requirements analysis and definition
* System and software design
* Implementation and unit testing
* Integration and system testing
* Operation and maintenance
* The main drawback of the waterfall model is the difficulty of accommodating change after the process is underway. One phase has to be complete before moving onto the next phase.
* Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.
* Therefore, this model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process.
* Few business systems have stable requirements.

The waterfall model is mostly used for large systems engineering projects where a system is developed at several sites.

**Evolutionary development**

* Exploratory development
  + Objective is to work with customers and to evolve a final system from an initial outline specification. Should start with well-understood requirements and add new features as proposed by the customer.
* Throw-away prototyping

Objective is to understand the system requirements. Should start with poorly understood requirements to clarify what is really needed.



* Problems
  + Lack of process visibility;
  + Systems are often poorly structured;
  + Special skills (e.g. in languages for rapid prototyping) may be required.
* Applicability
  + For small or medium-size interactive systems;
  + For parts of large systems (e.g. the user interface);
  + For short-lifetime systems.

**Component-based software engineering:**

* Based on systematic reuse where systems are integrated from existing components or COTS (Commercial-off-the-shelf) systems.
* Process stages
  + Component analysis;
  + Requirements modification;
  + System design with reuse;
  + Development and integration.
* This approach is becoming increasingly used as component standards have emerged.

**Reuse-oriented development**

4.3 Component-basedSE.eps                                      000FF8ECMacintosh HD                   B8AA5F2E:

**Incremental delivery**

* Rather than deliver the system as a single delivery, the development and delivery is broken down into increments with each increment delivering part of the required functionality.
* User requirements are prioritised and the highest priority requirements are included in early increments.
* Once the development of an increment is started, the requirements are frozen though requirements for later increments can continue to evolve.

4.4 Incremental-delivery.eps                                   000FF8ECMacintosh HD                   B8AA5F2E:

**Spiral development:**

* Process is represented as a spiral rather than as a sequence of activities with backtracking.
* Each loop in the spiral represents a phase in the process.
* No fixed phases such as specification or design - loops in the spiral are chosen depending on what is required.
* Risks are explicitly assessed and resolved throughout the process.
* **Objective setting**
  + **Specific objectives for the phase are identified.**
* **Risk assessment and reduction**
  + **Risks are assessed and activities put in place to reduce the key risks.**
* **Development and validation**
  + **A development model for the system is chosen which can be any of the generic models.**
* **Planning**
  + **The project is reviewed and the next phase of the spiral is planned.**
* **Objective setting**
  + **Specific objectives for the phase are identified.**
* **Risk assessment and reduction**
  + **Risks are assessed and activities put in place to reduce the key risks.**
* **Development and validation**
  + **A development model for the system is chosen which can be any of the generic models.**
* **Planning**
  + **The project is reviewed and the next phase of the spiral is planned.**

**4.5 Spiral-model.eps                                           000FF8ECMacintosh HD                   B8AA5F2E:**

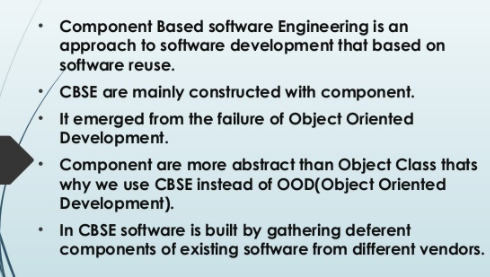
* Objective setting
  + Specific objectives for the phase are identified.
* Risk assessment and reduction
  + Risks are assessed and activities put in place to reduce the key risks.
* Development and validation
  + A development model for the system is chosen which can be any of the generic models.
* Planning
  + The project is reviewed and the next phase of the spiral is planned.

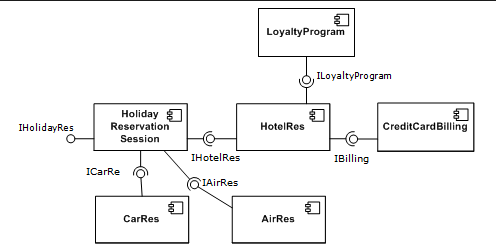
**5) Write notes on**

**a) Component based S/W engineering**

**b) Software re-use**

Component-based development (CBD) is a procedure that accentuates the design and development of computer-based systems with the help of reusable software components. With CBD, the focus shifts from software programming to software system composing.  
  
Component-based development techniques involve procedures for developing software systems by choosing ideal off-the-shelf components and then assembling them using a well-defined software architecture. With the systematic reuse of coarse-grained components, CBD intends to deliver better quality and output.  
  
Component-based development is also known as component-based software engineering (CBSE).





**Software re-use**

**What is software reuse?**

Software reuse is a term used for developing the software by using the existing software components. Some of the components that can be reuse are as follows;

* Source code
* Design and interfaces
* User manuals
* Software Documentation
* Software requirement specifications and many more.

## **What are advantages of software reuse?**

* Less effort
* Time-saving
* Reduce cost
* Less reuse
* Increase software productivity
* Utilize fewer resources
* Leads to a better quality software.

Reuse software engineering is based on guidelines and principles for reusing the existing software.

# ****What are stages of reuse-oriented software engineering?****

* **Requirement specification:**
  + First of all, specify the requirements. This will help to decide that we have some existing software components for the development of software or not.
* **Component analysis**
  + Helps to decide that which component can be reused where.
* **Requirement updations / modifications.**
  + If the requirements are changed by the customer, then still existing components are helpful for reuse or not.
* **Reuse System design**
  + If the requirements are changed by the customer, then still existing system designs are helpful for reuse or not.
* **Development**
  + Existing components are matching with new software or not.
* **Integration**
  + Can we integrate the new systems with existing components?
* **System validation**
  + To validate the system that it can be accepted by the customer or not.

