**Activity diagram** describe dynamic aspects of the system. Activity diagram is essentially an advanced version of flow chart that modeling the flow from one activity to another activity.

A **swimlane** diagram is a type of flowchart that delineates who does what in a process.

**Swim lane Diagram sometimes called a cross-functional diagram**

* It is a process flowchart that provides richer information on **who** does **what**.
* It can also be expanded to show times—**when** tasks are done and **how long** they take.
* Allows represent the flow of activities described by the use-case and at the same time indicate which actor or analysis class has responsibility for the action described by an activity rectangle.

A **class** is a description of a set of objects that share the same attributes, operations, relationships, and semantics.

**3 properties : Class Name, attributes, operations**

**Attribute** is a named property of a class that describes the object being modeled. In the class diagram, attributes appear in the second compartment just below the name-compartment

**Class diagram** is a static diagram. It represents the static view of an application. Class diagram describes the attributes and operations of a class.

**Operations** describe the class behavior and appear in the third compartment

An **interface** is a named set of operations that specifies the behavior of objects without showing their inner structure. It can be rendered in the model by a one- or two-compartment rectangle, with the stereotype

The **UML diagrams like activity diagram, sequence diagram** can only give the sequence flow of the application but **class diagram** is a bit different. So it is the most popular UML diagram in the coder community

The **purpose of the class diagram** can be summarized as:

* Analysis and design of the static view of an application.
* Describe responsibilities of a system.
* Base for component and deployment diagrams.
* Forward and reverse engineering.

**CRC Class responsibility Collaborator**

* **General classifications for a potential class**
  + **External entity** (e.g., another system, a device, a person)
  + **Thing** (e.g., report, screen display)
  + **Occurrence** or event (e.g., movement, completion)
  + **Role** (e.g., manager, engineer, salesperson)
  + **Organizational** unit (e.g., division, group, team)
  + **Place** (e.g., manufacturing floor, loading dock)
  + **Structure** (e.g., sensor, vehicle, computer)
* **Class Types**
  + **Roles** that people who use the system or that the system keep track of can play.
  + **Devices**: Sensors, motors, etc., which the system must monitor or control.
  + **Other** ``Things'' (physical objects) that the system must keep track (or an inventory) of.
  + **Properties**: Important characteristics or properties of part of the problem domain that the system must remember.
  + **Occurrences** or events, provided that the system must maintain a record of them.

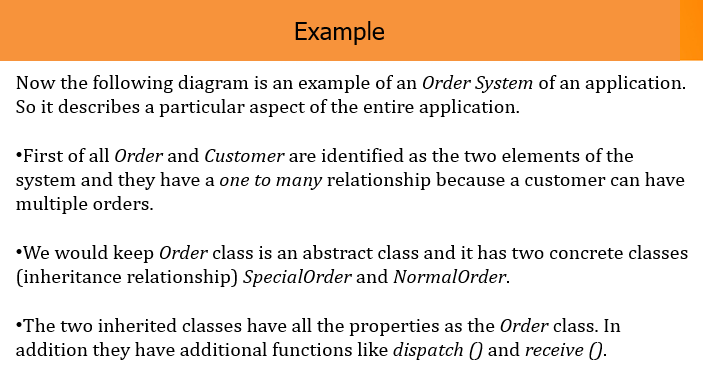
Structures, Assemblies, or Organizational Units.

**Class Characteristics**

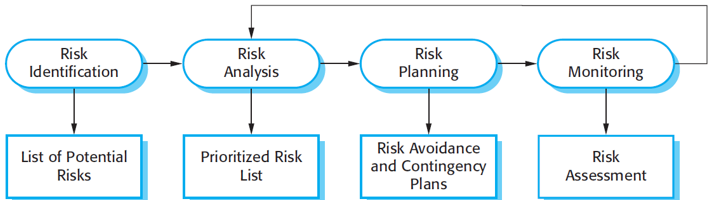
* + **Tangibility**: Does the class represent a ``tangible'' (physical) object, or is it more ``abstract''?
  + **Inclusiveness:** Is the class ``atomic,'' or is ``aggregate'' - that is, an assembly (so, that, it contains objects from other classes, as shown by a whole-part structure)?
  + **Sequentiality:** Is the class ``sequential,'' so that its operation is controlled by external changes, or is it ``concurrent,'' so that it behaves independently from other parts of the system - has its own thread of control?
  + **Persistence**: Is the class ``transient,'' so that its objects are all destroyed when the system quits, or is it ``persistent,'' so that its objects must be stored when the system quits and restored the next time the system starts up?
  + **Integrity**: Is the class ``corruptible,'' in the sense that it doesn't protect its internal resources from outside influence, or is it ``guarded,'' in the sense that its resources are protected?

Class diagram has three kinds of relationships.

* **Association** -- a relationship between instances of the two classes. There is an association between two classes if an instance of one class must know about the other in order to perform its work. In a diagram, an association is a link connecting two classes.
* **Aggregation** -- an association in which one class belongs to a collection. An aggregation has a diamond end pointing to the part containing the whole. In our diagram, **Order** has a collection of **Order Details**.
* **Generalization** -- an inheritance link indicating one class is a superclass of the other. A generalization has a triangle pointing to the superclass. ***Payment*** is a superclass of **Cash**, **Check**, and **Credit.** An end may have a **role name** to clarify the nature of the association. For example, an **OrderDetail** is a line item of each **Order**.
* A **navigability** arrow on an association shows which direction the association can be traversed or queried. An **Order Detail** can be queried about its **Item**, but not the other way around.
* The **multiplicity** of an association end is the number of possible instances of the class associated with a single instance of the other end. Multiplicities are single numbers or ranges of numbers. In our example, there can be only one **Customer** for each **Order**, but a **Customer** can have any number of **Orders**.

 **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;



**1.Technology risks** Risks that derive from the software or hardware technologies that are used to develop the system.

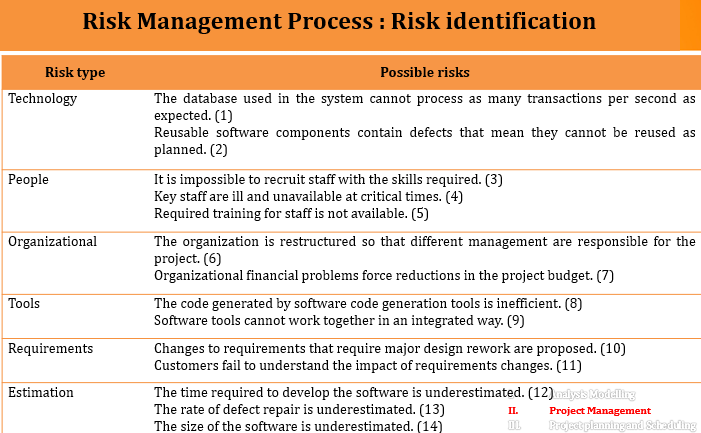
**2.** **People risks** Risks that are associated with the people in the development team.

**3.** **Organizational risks** Risks that derive from the organizational environment where the software is being developed.

4. ***Tools risks*** *Risks that derive from the software tools and other support software* used to develop the system.

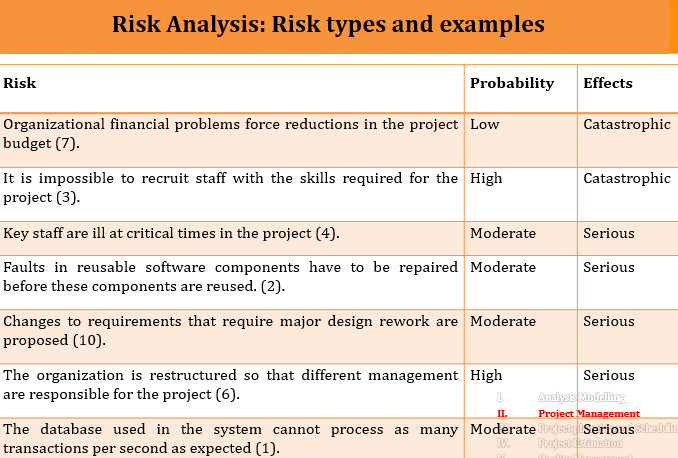
5. ***Requirements risks*** *Risks that derive from changes to the customer requirements* and the process of managing the requirements change.

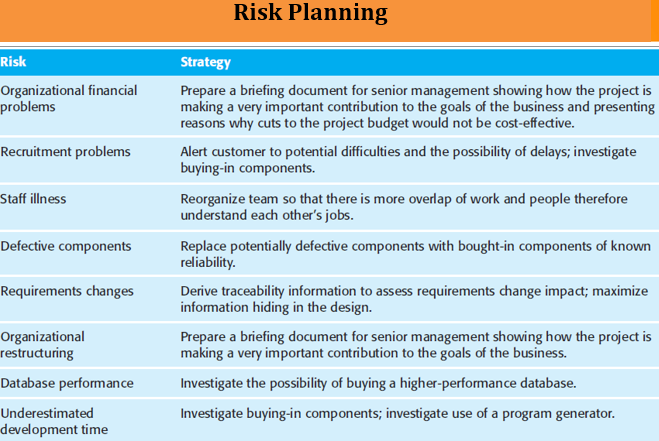
6. ***Estimation risks*** *Risks that derive from the management estimates of the* resources required to build the system.





* Assess probability and seriousness of each risk.
* The probability of the risk might be assessed as very low (<10%), low
* (10–25%), moderate (25–50%), high (50–75%), or very high (>75%).
* The effects of the risk might be assessed as catastrophic (threaten the survival of the project), serious (would cause major delays), tolerable (delays are within
* allowed contingency), or insignificant.





**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

**Project planning and Scheduling**

* Project planning involves breaking down the work into parts and assign these to project team members, anticipate problems that might arise and prepare tentative solutions to those problems.
* The project plan, which is created at the start of a project, is used to communicate how the work will be done to the project team and customers, and to help assess progress on the project.