

CS504 - Software Engineering - I Glossary By www.virtualians.pk

Abstract Class : In object-oriented programming, a class designed only as a parent from which sub-classes may be derived, but which is not itself suitable for instantiation. Often used to "abstract out" incomplete sets of features which may then be shared by a group of sibling sub-classes which add different variations of the missing pieces.

ACM : Association for Computing Machinery

Active object : Active object An object that encompasses its own thread of control.

Activity Diagram : Depicts high-level business processes, including data flow, or to model the logic of complex logic within a system.

Actors : An actor is a person, organization, or external system that plays a role in one or more interactions with your system. Actors are drawn as stick figures.

API : API (Application Program Interface)

APP : AAP The Association of American Publishers: engaged in standardisation efforts in document preparation

Association : Associations between actors and use cases are indicated in use case diagrams by solid lines. An association exists whenever an actor is involved with an interaction described by a use case. Associations are modeled as lines connecting use cases and actors to one another, with an optional arrowhead on one end of the line.

Byte : Byte A data unit of several bits smaller than a computer word: usually 8 bits.

Cache : Cache A small fast memory holding recently-accessed data, designed to speed up further access

CAUSE : CAUSE An international (mainly North American) nonprofit association for managing and using information technology in higher education

CMM : CMM (Software Capability Maturity Model)

Composability : Ability to compose systems from plug-and-play components

COMSOFT : COMSOFT Consortium for the Management of Emerging Software Technologies

COTS : COTS (commercial off-the-shelf software)

DLL : DLL (dynamic link library)

Evolvability : Support for new capabilities or ability to exploit new technologies

Functional Requirement : Functional requirements describe the functionality of the product. They describe exactly what tasks the software must perform. Functional requirements define the scope of the system, the product boundaries, and its connections to adjacent systems. Functional requirements also define the business rules. Business rules are the rules that the system must conform to, based on the individual business. This includes defining the data that must be tracked. The business rules are the most important type of functional requirements and most of your requirements will be of this type.

ISO : ISO (International Standards Association)

JAD : JAD (Joint Application Development)

Localizability : Ability to make adaptations due to regional differences

Modifiability : Ability to add (unspecified) future functionality

Non-Functional Requirement : Non-Functional requirements describe the look and feel of the system. This includes the visual properties of the system, its usability, and the performance requirements – how big, how fast, etc. Non-Functional requirements also include the product's intended operating environment and any maintainability, portability and security issues. Non-Functional requirements also include cultural and political issues as well as legal requirements that the software must conform to.

OOA : OOA (object oriented analysis)

OOD : OOD (object oriented design)

OOP : OOP (object oriented programming)

Quality : Quality is the degree of match between the product requirements (stated or otherwise) and the actual product. It is defined from the point of view of the user's perception, expectation and goals or need.

RAD : RAD (Rapid Applications Development)

Reusability : Ability to (re)use in future systems

SRS : The Software Requirements Specification (SRS) is a collection and organization of all the requirements surrounding a project. As the Vision Document was a broad statement of user needs, goals and objectives, and features of the system, the SRS begins the detailing of those needs and features, and how they are going to be implemented in the solution. The SRS is a collection, or package, of artifacts that describe the complete external behavior of the system

UCD : UCD (user centred design)

UML : UML (Unified Modelling Language)

Use cases : A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse.