Software Architecture

UNIVERSITY OF ALBERTA

Course Glossary

We have attempted to identify the most important terms in the course here. Look here if you encounter a term without a definition in context. If you need further explanation, you may have to return to the lesson where we first introduced the term, or do some independent study.

4+1 model view - a tool for looking at software from four different perspectives, and with use case, called **scenarios**.

abstract data types - data types that are defined by the user rather than the language. Their interactions are driven by their behaviour as opposed to their structure.

abstraction - simplifying a concept by ignoring unimportant details.

activity diagram - a flowchart for activities in a software.

artifact - a physical result of the development process, such as an executable or config file.

application engineering - the design and development of a specific product, using tools developed in domain engineering.

asynchronous - in request/response messaging, refers to when the client continues processing logic after sending a request. The response is dealt with separately.

ATAM - architectural tradeoff analysis method, a methodology for evaluating system architecture.

availability - the quality of being ready to perform its intended functions.

ball connector - part of a component diagram. Depicts a provided interface. Usually used by a **client**. Goes with a **socket connector**.

buffer overflow - when a buffer is being written and the data overflows its bounds, potentially overwriting adjacent memory.

client-host - a machine hosting a client-side process.

client/server - a type of relationship characterized by two roles: a client which requests services and a server which responds.

code reuse - the act of reusing parts of code in other systems.

compiler - a tool that converts higher-level code into machine language (in a binary file) and produces an **executable file**.

component diagram - a diagram that separates a software system into logical groupings (called components) with defined interfaces.

conceptual integrity - the coherency of a system, or the extent to which the parts contribute to the architecture of the whole.

Conway's Law - an observation that systems tend to emulate the organizational structure in which they were produced.

data integrity - the property of ensuring data is persistent and only modified in ways intended by design.

deployment diagram - a diagram that maps software components to **deployment targets**, either hardware or software executable environments.

data flow architecture - an architecture characterized by flows of data that are processed by filters.

data integrity - ensuring data is consistent and accurate over time.

data persistence - the concept that data persists beyond the execution of a program which uses the data.

database - software that stores and serves data.

deadlock - a situation in which a process is halted while waiting on some other operation in the system, which in turn is waiting on another process.

decomposition - breaking a problem down into distinct components.

deployment target - often a **node**: the targeted location of a component of the software system.

designers - developers responsible for architectural design.

development view - a view of software components with respect to development resources, tools, or process.

domain engineering - the design and development of the general parts of a product line or product family.

encapsulation - bundling data and methods into an object and exposing the desired interface to other components.

feedback loop - a common form of **process control** in which a process is monitored and controlled based on the variable monitored.

feedforward control - a process control technique in which data from an upstream process is used to make decisions in a downstream process.

filter - in data flow architecture, a filter represents some data processing unit. It takes input from upstream filter(s) and outputs to downstream filter(s).

generalization - allowing for more use-cases by factoring out conceptual commonalities and abstracting them.

hypothesis and test - a method of testing that software fulfills its functions.

interoperability - being able to exchange information with external systems.

interpreter - a tool to translate higher-level code into machine language on-the-fly.

language translation architecture - various strategies for converting high-level language to binary code.

latency - the time it takes to produce an output after receiving an input.

layered architecture - an architecture characterized by arranging the separation of responsibilities into layers.

logical view - a view focusing on the functional requirements of the system or a component of the system. Often Class diagram or State diagram.

machine language - the ultimate form of any instruction sent to a device. In the form of binary (ones and zeros).

maintainability - the property of being easily changed.

manifestation - an abstraction relationship wherein an artifact manifests the behaviour of one or more components.

modifiability - a measure of the system's ability to change, including changes to functions, new functionality, or removing functionality.

multi-tier architecture - see n-tier architecture.

n-tier architecture - an architecture characterized by separating responsibilities into tiers; similar to layered architecture.

node - hardware or software execution environment such as an operating system or runtime environment.

object code - code produced by a compiler that the machine can understand.

overhead - computational resources that are used to enforce architecture, rather than the desired activities. Considered a **tradeoff**.

peers - as opposed to designers, members of the project who were not involved in architectural design.

performance - how well the system responds to commands, errors, and internal events, usually measured by **throughput** and **latency**.

physical view - view focusing on physical deployment of the system.

pipe - in data flow architecture, a pipe represents the passing of data from one filter to the next.

process control - controlling a physical process or system, usually done with software.

process view - a view which focuses on non-functional requirements, usually quality requirements such as availability and concurrency.

product line/product family - a group of products that share characteristics. Can be taken advantage of by developing their commonalities simultaneously.

quality attribute - characteristics of a software system that can be monitored and provide information about the overall quality of a system.

quality attribute scenario - a test designed to examine whether a software system meets quality attribute requirements.

reference architecture - the architecture framework that is planned out for a product family and filled-in, or adapted, for specific products.

request/response - a pattern of messaging in which a client sends a request and a server returns a response.

reusability - the quality of being reusable in different systems.

scenario - a specific use-case shown with the 4 views. Used to validate the views.

script - a formal description of a scenario. Utilizes the views to explain how the scenario is executed.

security - a measure of the system's resistance to unauthorized access and use.

separation of responsibilities - on a software architectural level, sorting components into groups with specialized functions.

server-host - a machine hosting a server-side process.

socket connector - part of a component diagram. Depicts a required interface. Often connected to a server process. Goes with a **ball connector**.

SQL - structured query language, a common language for managing a database.

subroutines - pieces of logical code called by the main program.

synchronous - in the request/response, this refers to when the client halts processing until receiving a response from the server.

testability - how easy it is to demonstrate errors in the system through executable tests.

throughput - the amount of output produced over time.

tradeoff - the result of design decisions: implies that there are benefits and detriments to every design decision and these must be weighed against each other.

usability - the degree to which the system is easy to use by end-users. Could include intuitiveness, number of user errors, useful user feedback, and other metrics.

virtual machine - an emulation of a computer system.