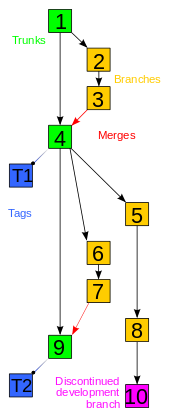
**Codebase**: In software development, a codebase (or code base) refers to a whole collection of source code that is used to build a particular software system, application, or software component. Typically, a codebase includes only human-written source code files. A codebase is typically stored in a source control repository that belongs to a revision control system.

**Repository**: In revision control systems, a repository[1] is a data structure which stores metadata for a set of files or directory structure. Depending on whether the version control system in use is distributed (for instance, Git or Mercurial) or centralized (Subversion or Perforce, for example), the whole set of information in the repository may be duplicated on every user's system or may be maintained on a single server.

**Version**: Software upgrade versioning is the process release of assigning either unique version names or unique version numbers to unique states of computer software. Within a given version number category (major, minor), these numbers are generally assigned in increasing order and correspond to new developments in the software. At a fine-grained level, revision control is often used for keeping track of incrementally different versions of information, whether or not this information is computer software.

**Revision**: A component of software configuration management, version control, also known as revision control or source control,[1] is the management of changes to documents, computer programs, large web sites, and other collections of information. Changes are usually identified by a number or letter code, termed the "revision number", "revision level", or simply "revision". For example, an initial set of files is "revision 1". When the first change is made, the resulting set is "revision 2", and so on. Each revision is associated with a timestamp and the person making the change. Revisions can be compared, restored, and with some types of files, merged.

**Tekening**1. Merge: twee verschillende versies van dezelfde code integreren.  
2. Tag: een belangrijke versie van de codebase benoemen, zodanig dat die later makkelijk terug kan opgehaald worden.   
3. Branch: een codebase dupliceren om op beide versies apart door te werken. Elke branch wordt onafhankelijk verder geupdate. Branches kunnen na verloop van tijd terug gemerged worden.   
4. Trunk: er kunnen verschillende branches naast elkaar bestaan, maar meestal is er ook een branch die de “basisversie” van de software is. Dit noemt men dan een trunk – de hoofdversie van de code.

**Uitleg wikipedia tekening**

In terms of graph theory, revisions are generally thought of as a line of development (the trunk) with branches off of this, forming a directed tree, visualized as one or more parallel lines of development (the "mainlines" of the branches) branching off a trunk. In reality the structure is more complicated, forming a directed acyclic graph, but for many purposes "tree with merges" is an adequate approximation.

Revisions occur in sequence over time, and thus can be arranged in order, either by revision number or timestamp.[note 2] Revisions are based on past revisions, though it is possible to largely or completely replace an earlier revision, such as "delete all existing text, insert new text". In the simplest case, with no branching or undoing, each revision is based on its immediate predecessor alone, and they form a simple line, with a single latest version, the "HEAD" revision or tip. In graph theory terms, drawing each revision as a point and each "derived revision" relationship as an arrow (conventionally pointing from older to newer, in the same direction as time), this is a linear graph. If there is branching, so multiple future revisions are based on a past revision, or undoing, so a revision can depend on a revision older than its immediate predecessor, then the resulting graph is instead a directed tree (each node can have more than one child), and has multiple tips, corresponding to the revisions without children ("latest revision on each branch").[note 3] In principle the resulting tree need not have a preferred tip ("main" latest revision) – just various different revisions – but in practice one tip is generally identified as HEAD. When a new revision is based on HEAD, it is either identified as the new HEAD, or considered a new branch.[note 4] The list of revisions from the start to HEAD (in graph theory terms, the unique path in the tree, which forms a linear graph as before) is the trunk or mainline.[note 5] Conversely, when a revision can be based on more than one previous revision (when a node can have more than one parent), the resulting process is called a merge, and is one of the most complex aspects of revision control. This most often occurs when changes occur in multiple branches (most often two, but more are possible), which are then merged into a single branch incorporating both changes. If these changes overlap, it may be difficult or impossible to merge, and require manual intervention or rewriting.

In the presence of merges, the resulting graph is no longer a tree, as nodes can have multiple parents, but is instead a rooted directed acyclic graph (DAG). The graph is acyclic since parents are always backwards in time, and rooted because there is an oldest version. However, assuming that there is a trunk, merges from branches can be considered as "external" to the tree – the changes in the branch are packaged up as a patch, which is applied to HEAD (of the trunk), creating a new revision without any explicit reference to the branch, and preserving the tree structure. Thus, while the actual relations between versions form a DAG, this can be considered a tree plus merges, and the trunk itself is a line.

In distributed revision control, in the presence of multiple repositories these may be based on a single original version (a root of the tree), but there need not be an original root, and thus only a separate root (oldest revision) for each repository, for example, if two people starting working on a project separately. Similarly in the presence of multiple data sets (multiple projects) that exchange data or merge, there isn’t a single root, though for simplicity one may think of one project as primary and the other as secondary, merged into the first with or without its own revision history.