SDEVEN Software Development & Engineering Methodology

Version: 7.0.6
Release date: 230715

Conventions & Principles (SDEVEN.80-COPRI)

Table of Content

Preliminaries

This section is about terminology and naming standards and conventions used in development process and code.

The company try to keep aligned to international practiced and most used terms & conventions and adopts standard changes "on the fly" as soon as possible.

Is necessary that people that are working in software structures (departments) of company to *keep aligned* with terms and conventions in order to have **A COMMON LANGUAGE AND UNDERSTANDING ABOUT THINGS**.

The idea is that when is needed to write things that will be used by someone else in the other process steps or in the future after a while (ie, there are more that one people involved), it is important that **shared things to be recognized** at destination as they was thought at origin by those team that produced them.

Files

- files whose names start with `xxx` (case is not important) are (and will be) considered marked for a ***future deletion or discontinuing***'

why don't delete them directly

- this method allows to "remember" to delete them latter but to still keep the information available a while to assure an acceptable and as smooth change management (and to give time for a possible return to the previous situation)
- also when view list of files in alphabetically order, these files appear grouped near the end (or beginning)

- files whose names start with `_WIP` or `WIP` (normally case doesn't matter but should be a sign to pay more attention) are known (announced) as ***being in work and not in a stable state***, so they should be treated more carefully when need to use them
- any other "traditional" conventions should be respected and treated in consequence; the most of them comes from `Linux` systems, for example files beginning with dot (`.`) are hidden (for normal users), backup files have extension `.bck`, files with extension `.tmp` are temporary and subject to be deleted (by users or operating system) without notice, and so on
- the characters `-` (dash) and ` ` (space) will be avoided as much as possible in file names (ie, in different programs should be source of errors by confusing with arithmetic minus operator) and replaced with `_` (underscore); if this is not possible, A WARNING must be stated inside the file content or in a respetive component README file

Calendar dates

These should respect the convention as they will be written as `YYYYMMDD` or `YYMMDD`, because by doing so will assure a right ordering "by date" in about all situations (just by using the operating system standard sorting procedures and not requiring some special order methods); the year could be only from 2 characters if there is no doubt regarding the year (the standard conventions stated by *SQL ANSI* are very clear and self explanatory)

Datastores

The preferred datastore for code is the repository.

On the other hand, there are cases when *other type of stores are required*. Generally these stores are required in development process (stores for other processes are not subject if this methodology) and specific to project. They can be permanent (as stores where kits resides or stores for sales materials) or temporary allocated for project.

In any cases the `dev infrastructure admin` should be contacted.

Normally these stores will be allocated from file server pools, which means that will act as "simple shared drives" (without any notice). If there are some other protocols required, such as access by `http`, `https`, `rsync` and so on, *these issues must be notified*.

Also you should always expect that for such stores there are some limitations such as maximum capacity allowed, number of files, lifetime of files, file foemats allowed, etc. If this could be issues or you concern about them, please ASK and do not make other assumptions.

"In code" names & identifiers

These issues should follow the programming language standards (as `PEP` for `Python`) or best practices in case there are no stated standards. A *linter* and / or code formatter should be used like `Blake` for `Python`, but better is to *ASK* the team leader or project technical manager and NOT TO USE your own standards by supposing they are good and should be used (if this is the case, please discuss this with technical stuff before putting it in practice).

Some of recommended practices in any cases are:

- always mark protected or private attributes with underscore character in front of their names, regardless the programming language used
- always comment the code; as frequently as better; do not worry about readability or other concerns; somebody will take care to ask for cleaning if seems to be too much "spam"
- use UPPER CASE for identifiers used or intended to use as constants
- comment the functions or class methods with a large comment block and specify at least: a description of max 2 lines, the argument types and what are good for, the returns type and when is happening

---

Last update: July 18, 2023