SDEVEN Software Development & Engineering Methodology

Version: 7.0.6
Release date: 230626

Licenses and Products (SDEVEN.70-LIP)

Table of Content

- Licenses and Products (SDEVEN.70-LIP)
  - Preamble
  - Software licenses universe
  - Software licenses
  - System vs Product
  - Software products / models
    - By completeness taxonomy
    - By code level taxonomy

Preamble

This section refers to software licenses and products, how are used and what are good for.

Procedure present what kind of licenses are practiced and generally how a software system becomes a product (or when can be considered product).

This procedure is not mandatory for software development process itself but gives an idea about "what will happen with product for which I did write code". The licensing problem is a commercial and legal one. Its just "for your information", just because many people want to know what becomes their work, and because sometimes is good to know why some decisions are made and wht is the rationality behind them.

Software licenses universe

A software license, without any other clause, is about the legal(ity) and right(s) to use the respective product / system.

Generally speaking, you can download, copy, install, remove, etc the system, free and in most cases without any payment required. All legal.

But there are two things that are not always free and MUST BE ESTABLISHED BY LICENSE:

- A FUNDAMENTAL FACT: The one **you are not the proprietary of code**, you cannot treat it as its your own property, but you can use it ***JUST for your own and IN YOUR NAME***, make safety copies, backup it, restore it, and
- You **cannot use** it without permission (from legal point of view), EVEN THEN SYSTEM DOES NOT ENFORCE ANY RESTRICTION

Software licenses

There are 3 kind of license models (types) used by company for software products:

- **open licenses** - these are free (there is no cost to pay for them); for these licenses the company still preserve the *intellectual property and copyright*, just offer for free the rights of use; any other services *are not necessarily and default covered by this license*, but if you how to do it, you're free to download the software, install and use it without any obligations; however, ***you cannot sale / resale*** the system /product as it would be your own property
- **turn key licenses** - these are systems / products made especially for a customer, eventually paid by him and the intellectual property is transferred to the customer; so, after finishing the system, ***company has no right to sell / resell or use the code or parts of code AS IT WAS TRANSFERRED***; however, in most cases, two "evidence" read only CD are made with the code for which the intellectual property is transferred
- **commercial licenses** - these are strictly with payment for usage; quantification of payment is / can be made in various forms, for example number of users, number of computers, number of processors, quantity of memory, and so on; in most cases, the software restrict usage by "forcing" in a way these quantities, of course first thing being the software ability to "count" for them

Of course there are many other taxonomies but these are relevant for software development as activity, the others being useful for other domains like product management or legal / lawyers structures.

> **IMPORTANT notice**: If not otherwise specified, open licenses should be accompanied with text: "**This software is a copyright of company Systems (REN CONSULTING SOFT ACTIVITY SRL).**". Text should be put at start of license content as to not alter its original text which is usually published and can be referred AS IS.

System vs Product

For a software there are 3 major "end of cycle": to become a product, to become a system or both.

As **system**, a software should have an installation procedure, which could be just a documentation or other automation software. This procedure should be *clear*, *well defined* (ie, deterministic, without ambiguities) and *repeatable*.

As **product** must have an *usage documentation* (ie, day by day), an *administration documentation* (ie, installation, configuration, maintenance), a *packaging* procedure.

These are from manufacturing point of view, other processes requiring any other issues, for example in marketing a logo would bw required for a product, etc.

This methodology assures that the essential parts of both taxonomies will be covered, at least in raw forms creating the base for future / next refinement levels.

Software products / models

From this point of view relevant taxonomies are:

- by completeness
- by code level

By completeness taxonomy

- **full standalone** - products that contains everything to assure a complete functionality (aka full stack products, or in jargon "with batteries included", all in one, etc)
- **modules** - products that assure a single functionality usually useless only itself, but normally used in a large context, combined with other modules; examples: a database JSON transformer, a caching system, a queuing systems, etc
- **frameworks** - products aimed to be used as foundation to build other products over it; examples: python Flask, company CORE, etc
- **interfaces** - products aimed to "stay in front" of other systems / products and therefore assuring different kind of protection, translation, etc; often known as middleware products; examples: data APIs, proxies, guards, data translators, SQL Alchemy, etc

By code level taxonomy

- **low level / infrastructure** - these are systems that address low level operations, with intensive (ie, directly coded) use of operating system directives; examples: print utilities, file system watchers, system monitors, serializer, de-serializer, en(de)coders, etc
- **mid and high level** - these are systems that do not address directly operating system directives (just in rare cases for usual file operation), usually addressed to business or just to assets inventory (infrastructure systems for example); examples: ERPs, invoice makers, etc
- **UI / meta** - these are systems that assure some features for user interface (operations) by using different flavours of (tagging) languages specific to a device (for example VT100 terminals), to a software (for example HTML for browsers); sometimes these systems use "real" languages with empowerment of complex programming languages (for example JavaScript) or just simple "stylers" to assure a better readability (CSS is a good example, Markdown and PostScript are others, etc)

Last update: July 18, 2023