Semantic Versioning

concept cards©
FOR TEAMS, MANAGERS OR LEADERS

Everything should have a meaningful version number.

How

In semantic versioning, version numbers obey to the following structure:

MAJOR.MINOR.PATCH

MAJOR counts upwards for significant or structural changes and for major feature or functional enhancements.

MINOR counts upward for minor feature or functional enhancements or less significant content changes. (For software, minor changes are backward compatible).

PATCH counts upwards for corrections or bug fixes. No features or functionalities are added and content is not changed.

According to this logic, users of the versioned thing can immediately see that it has been changed and how significant the change was. As a result, they can further review the changes to evaluate their impact and consequences.

Why

The larger the dependencies the more important it is to know if changes have been made to an items and if these changes will need changes elsewhere.

Even though versioning is absolutely indispensable in the IT and technical world it has not made its way into most of the commercial world, e.g. into versioning presentations, papers, texts, organizations, processes, concepts, ideas, products, etc. Semantic versioning would also add value here, especially if the context is also complex, fast-changing or agile.

Semantic Versioning represents a way of thinking. The current version is the best at the moment but will certainly be improved.

Add-ons

After MAJOR.MINOR.PATCH further information can be added in dot-notation, e.g. statuses, initials, languages, extension: v2.4.1.beta.ua.en.pdf