Sprint Evaluation

Table of Contents

Assessment and rating of sprints

This section explains my way of evaluating sprints in most of my project-based teachings.

A TA evaluates each Sprint (one per week, most of the time). The evaluation will address five or six criteria and will take the following form:

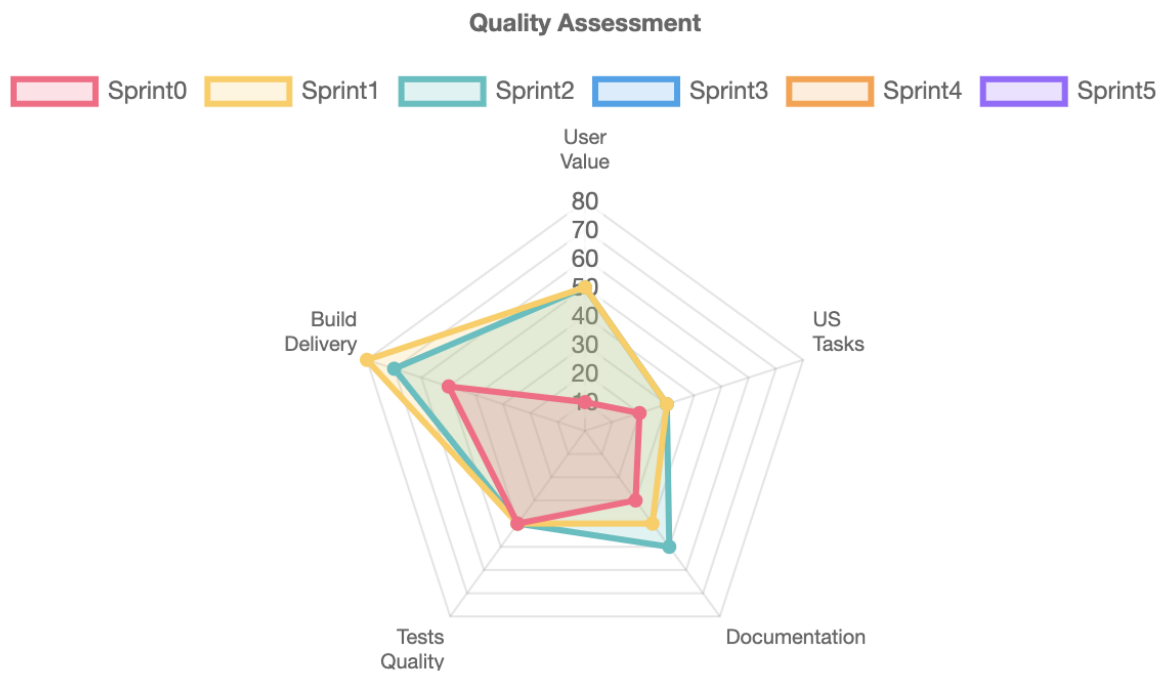


Figure 1. Example of weekly evaluation (using JS)

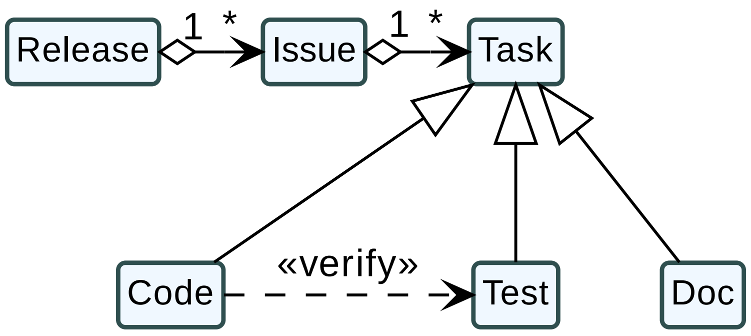


Figure 2. The initial 6 artifacts (Source here)

These artifacts come from a course on software quality from my colleague Xavier Blanc (https://github.com/xblanc33/QualiteDev).

Sorry for the French in the linked explanations.

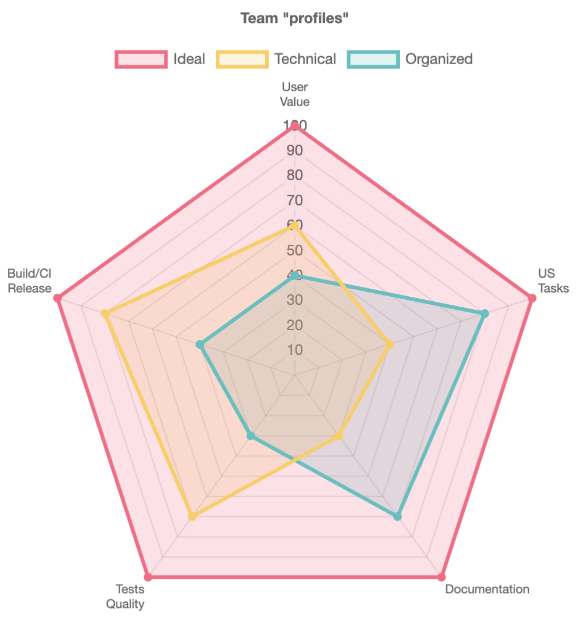


Figure 3. Examples of typical team profiles

User values

- The way the goals have been captured (more details here)
- Evaluated by the *Product Owner*
- Should never decrease
- Sometimes no, or low, added value (Spikes, refactoring sprints)

US/Tasks

- The way Goals, US, and tasks are linked and traced here)
- Very tool-dependent (e.g., blocking issues, task lists)
- Can (should?) reach a high level pretty early

Documentation

- Technical and user documentations (more details here)
- As much automated as possible (javadoc, `.md`/`.adoc`, code included rather than copy-pasted)

Tests/Quality

- How well are supported/explained the verification activities (more details here)
- Address and differentiate unit tests and integration tests

Build/CI/Release

- How professional and automated are the build, automated testing, deploy (more details here)

- Can (should?) reach a high level pretty early

Project typical evaluation sheet

Here is a typical scale:

Criterion	**%**
Respect for the Scrum method	20%
"Professional" character of dev	20%
Successive deliveries	20%
Tests / Documentations / Readme / wiki	20%
Code and application quality	10%
Final Customer Satisfaction	10%

I advise you to add such a table in your readme and self-evaluate your project.

Useful tips

Technical Debt

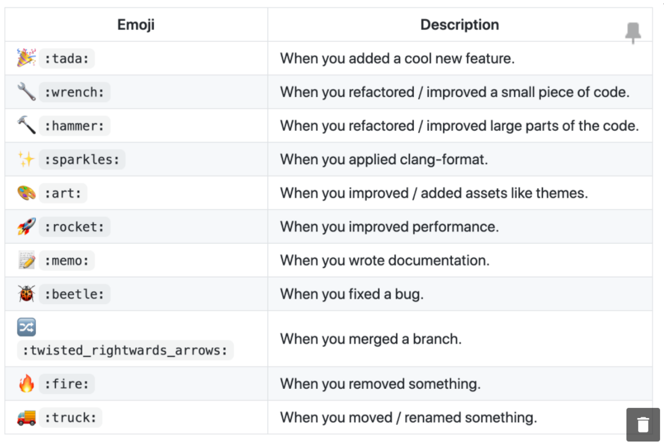
Software Engineering term for *Procrastination*!

technical debt 45min

Commit messages

Have the same policy in the project:

`[Fix|Feature|⋯] [Issue_Number]: Use a sentence with a capital letter and verb for the first word.`



Emoji	Description
🎉 :tada:	When you added a cool new feature.
🔧 :wrench:	When you refactored / improved a small piece of code.
🔨 :hammer:	When you refactored / improved large parts of the code.
✨ :sparkles:	When you applied clang-format.
🎨 :art:	When you improved / added assets like themes.
🚀 :rocket:	When you improved performance.
📝 :memo:	When you wrote documentation.
🐞 :beetle:	When you fixed a bug.
🔀 :twisted_rightwards_arrows:	When you merged a branch.
🔥 :fire:	When you removed something.
🚚 :truck:	When you moved / renamed something.

Use badges

Checks passing | Tests passing | code quality B | License MIT

Comments in code

Avoid useless comments!

Figure 4. (source : https://pic.twitter.com/ICGb9qKnRN)

Useful links

- General
  - The materials for the course: http://bit.ly/jmb-teaching
  - The initial course about quality development: https://github.com/xblanc33/QualiteDev
- Python
  - Defect prediction https://github.com/awsm-research/PyExplainer