

Software Development Proficiency

Each level of achievement builds on the previous level, from left to right.

Proficiencies (weight)	Below Expectations	Meets Most Expectations	Exceeds Expectations
Version Control (10)	Few commits are made, often addressing multiple concerns (not atomic).	Each commit addresses just one project concern (atomic).	Each commit message succinctly describes the purpose of the commit.
Issue Management (10)	Few issues are made, with limited comments, labels and milestones.	Effectively uses issues to organize the work, including labels and helpful comments.	Issues are linked to commits via the commit messages, and milestones are used to further organize the work.
Readability: Layout (5)	Contains many long lines of code or blocks of commented-out code which should have been deleted.	Lacks consistent and appropriate indentation/spacing rules.	Code has consistent and appropriate indenting and spacing that reflects the logical structure of the code.
Readability: Logic (10)	Most names used are short and unhelpful.	The naming of code elements does reflect their purposes.	Modules/functions/variables are clearly named to reflect their functionality (self-documenting code).
Project Structure (5)	The code consists of a few very large files containing disparate elements.	The code is comprised of separate, well-focused files, none of them too large.	The project as a whole is organized in a sensible file structure.
Architecture (5)	There is limited or no description of the overall architecture and module interactions.	There is a document adequately describing the architecture and module interactions	There is an extensive description of the architecture and module interactions using a modelling language such as UML.
Code Modularity (10)	The code is barely decomposed into modules/classes.	The code is decomposed into suitable modules with well-defined responsibilities.	The role and function of each module is clearly documented in the module comments.

Proficiencies (weight)	Below Expectations	Meets Most Expectations	Exceeds Expectations
Maintainability: Data Modelling (5)	Data is stored in individual variables without using appropriate data structures to organize it.	Common data structures such as arrays and objects are appropriately used to organize the project data.	The data representation model and reasons for the corresponding data structure choices are clearly described in a suitable document.
Maintainability: Functional Decomposition (10)	The project has frequent code duplication and functions without a clear, singular purpose.	Code duplication is usually avoided by use of functional decomposition.	Each function has a clear, singular purpose. Long functions are avoided.
Maintainability: Testing (10)	No visible automated tests.	Automated tests are present but occasionally insufficient.	Has adequate automated unit testing code for public functions.
Maintainability: Resource Separation (5)	Special constants and string literals are used directly in the program logic.	Special constants and string literals are named and are separated from the program logic.	Data resources are stored in a suitable data format and separated from the business logic of the project.
Code Documentation (10)	The code contains limited or unhelpful comments.	The modules and functions are adequately documented in comments; inline comments are present where needed.	Module and function documenting comments are comprehensive and follow the documentation standard for the given language.
User Documentation (5)	No user documentation.	Includes installation instructions and very basic user's guide.	Includes an extensive user's guide in an appropriately sharable form (e.g., webpage).