ENGR 301 Performance Portfolio 2

Status: Draft version subject to amendment prior to finalisation.

This assessment is your opportunity to show your work, demonstrating how you have followed good project management practices over the duration of the project. You will be assessed on how well your individual portfolio shows you followed the processes and practices described in the course methodology in both the management and practical aspects of your project. This is an evidence-based assessment which will draw on one primary source: GitLab.

The assessment will be on a 4-trait basis:

1. Planning
2. Construction
3. Integration and Testing
4. Review and Feedback

These traits are an Engineer's core competencies, as described in the IEA Graduate Attributes and Professional Competencies (PDF), the GitLab Workflow overview, the DevSecOps Lifecycle and the Introduction to GitLab Flow documents.

Portfolio Format: Your portfolio must contain one section per trait and in each section you must:

1. Write 1 paragraph describing how you contributed to the trait over the assessment period, citing references in-line as evidence to support any claims.

2. Provide a numbered reference list of no more than six hyperlinked references to GitLab Issues, Comments, Milestones, Merge Requests, etc.,

References should be cited as: "some claim [1]".

Sources of Evidence: the GitLab Project is the primary source of evidence for the performance portfolio. Contributions to project wikis and Mattermost chat are secondary sources of evidence and will carry less weight in the portfolio. Other sources are out-of-scope as evidence without prior arrangement.

1. Planning

Highest Level: Work is directed toward documented well-defined outcomes and deliverables throughout the assessment period; project management knowledge areas **scope**, **time**, **cost** and **quality** have a well-organised and well-maintained representations in GitLab. The planning aspects of the course methodology are adhered to consistently.

Individual Portfolio

Assessment will be made of the evidence showing engagement with, and contribution to planning and organisation of work **scope** and **time** using Issues, Comments, Labels, Boards and Milestones, including:

- Issues raised to define work *prior* to the work being undertaken (rather than at the time, or even after, work has begun);
- Issues representing large bodies of work are broken down to actionable tasks and grouped by Milestones;
- Issue descriptions and comments contain an appropriate level of written detail;
- start and end/due dates are set for Issues and Milestones, which fit together to form a coherent whole;
- referencing what's related by using GitLab's cross-linking and referencing features.

2. Construction

Highest Level: Good practice, defined by the course methodology, is established and followed throughout execution.

This is your opportunity to show how you have followed good technical practice by following GitLab Flow which is a core part of the course methodology. You may find GitLab's Value Stream Analytics helpful in framing this section.

Individual Portfolio

- Git branching strategy adheres to GitLab Flow.
- Branches are created and linked to Issues via GitLab's automation features.
- Branches address a single goal or deliverable, i.e. a single issue.
- Commits are directed toward achieving the branch's goal.
- Commits are atomic: "a logically indivisible part" *or* "a single irreducible unit or component" of the branch's goal or deliverable.
- Commits conform to agreed standards.
- Code refactoring is a purposeful activity.

3. Integration and Evaluation

Highest Level: Technical work continuously integrated; regular contributions are spread evenly across the assessment period. Automated tests are created and extended; evaluations of existing work are performed.

Individual Portfolio

- Work is committed regularly (rather than near the end).
- Merge Requests are created at the same time as branches and linked to issues via GitLab's

automation features.
- Merge Requests are directed toward a single goal, i.e. a single issue.
- Pipeline failures of pre-commit are *extremely rare* due to local installation of pre-commit.
- Failed pipelines are *rare*: they are corrected immediately, not just before branch merge, and the underlying cause of failure is identified and fixed.
- Merge Requests are short-lived on the time scale of an iteration; there are no stale branches nor branches which are behind/ahead of `main` by several 10's of commits or more.
- Merge Requests form a *coherent* body of work when merged.
- Contributions are made to test suites and test suites are maintained over the assessment period.
- Evaluations are built iteratively and are reproducible.
- Data meeting evaluation criteria (including quality metrics) are integrated into the project as they are measured.

Please note: reviews of Merge Requests should take place *after* automated tests have passed (see the next section).

4. Review and Feedback

Highest Level: Experience and expertise is shared within a self-organising, cross-functional team with explicit collective code/system ownership, creating a positive feedback loop for iterations.

In a high-functioning team, at the beginning of an iteration, individuals are expected to self-select tasks, then get on with it. Tasks would be chosen to increase exposure to other parts of system, and to help up-skill other team members. Ideally, at the end of an iteration a documented *sprint* or *iteration review* takes place prior to planning for the next iteration. These reviews are for the purpose of sharing the tools, discoveries, and lessons learned during the iteration and to provide context for the planning of future iterations (plural!) *and* the project roadmap.

Individual Portfolio

- Reviews the work of others and their own work is reviewed by someone other than the author(s). **Note:** this requires evidence of engagement with the review process, i.e. written discussion in Merge Requests of potential errors, improvements, etc.
- Commits, contributions and reviews to/of multiple parts of the project, as seen in the various logs (git, Issue, Contribution, etc.).
- Individuals are observed to be team players and work as a whole team.
- Sharing tools, discoveries, and lessons outside of the team; helping members of other teams with their technical and non-technical work including facilitating retrospectives.

Author: James Quilty
Affiliation: School of Engineering and Computer Science, Victoria University of Wellington
Last updated: 2024-02-29