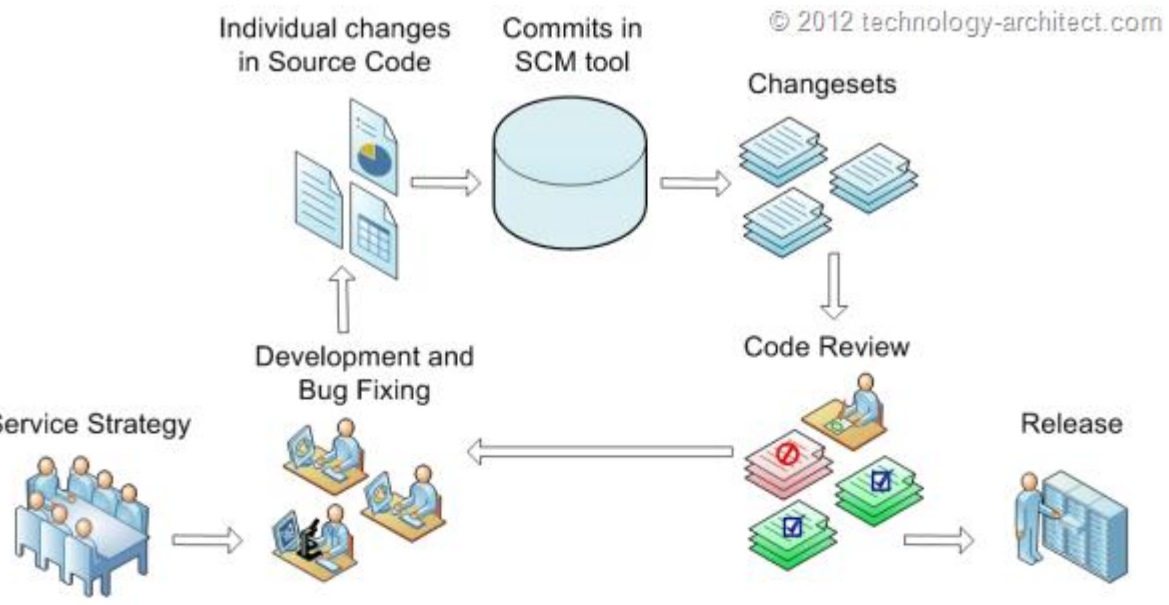
Code Review


© 2012 technology-architect.com
Individual changes in Source Code
Commits in SCM tool
Changesets
Code Review
Release
Development and Bug Fixing
Service Strategy

SWEN-261
Introduction to Software Engineering

Department of Software Engineering
Rochester Institute of Technology

Software Engineering
Rochester Institute of Technology

A code review can improve the quality of the product and the quality of the team.

- Increase product quality
  - ***Identify and fix design or coding violations***
  - ***Identify and fix code communication issues***
  - ***Analyze test coverage, identify new test scenarios***
- Increase overall team skill
  - ***Discuss code communication***
  - ***Share coding and testing techniques***
  - ***Discuss design principles & patterns, as appropriate***

There are several situations that warrant a code review.

- For new members of the team
  - *Along with reading the Design documentation*
  - *Code review (walk-through) with a senior developer*
- For Spikes
  - *To impart lessons from the Spike to the rest of the team*
- For User Stories
  - *To improve the quality of the feature code*
  - *To share best practices with the rest of the team*
  - *Even trivial stories should have reviews*

There are several code review techniques.

- Individual
  - *A senior developer sits with a junior developer*
  - *The review can be focused on a specific problem or for general understanding a subsystem*
- Synchronous
  - *A team meets to review some code*
  - *Usually the most formal process*
  - *Disadvantage of needing to sync schedules*
- Asynchronous
  - *A developer uses an online tool to create a review*
  - *Shows the diffs between two branches*
  - *Reviewers make comments in the tool*
- Hybrid approaches

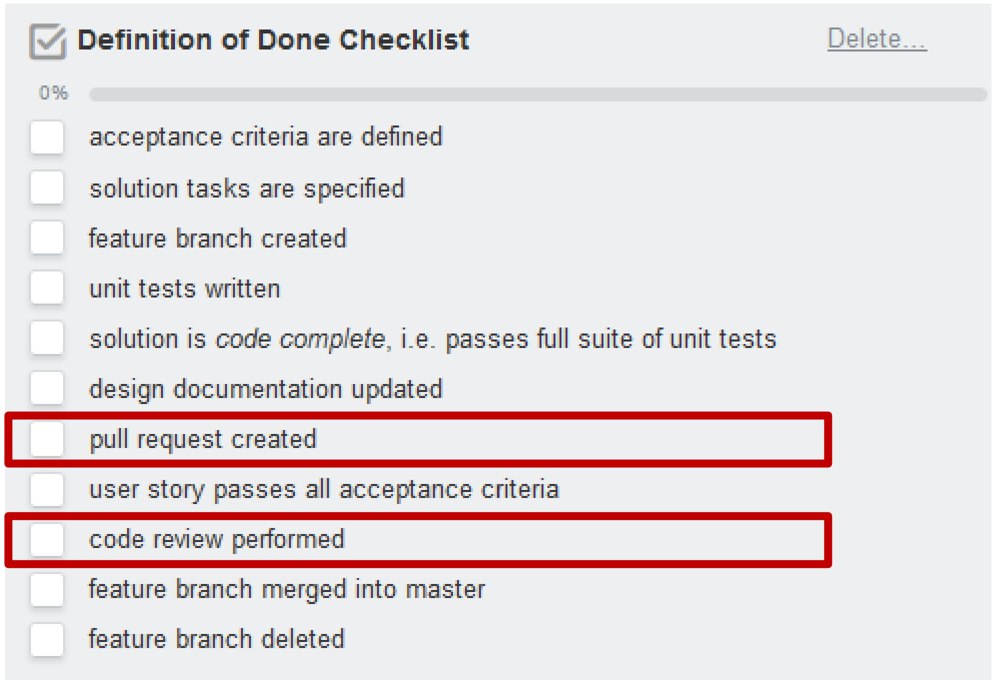
A team will often have a checklist of things to look for during the code review.

- Coding practices
  - *Code communication*
  - *Defensive programming practices*
- Design practices
  - *Adherence to architectural tiers*
  - *Adherence to core OO principles*
  - *Adherence to OO design principles*
- Testing practices
  - *Are test suites comprehensive (enough)*
  - *Test code follows good code and design practices*
- Design documentation
  - *Is the documentation being kept up-to-date*

The activity will guide the team through doing an asynchronous review.

- You will create a git *pull request* for a selected feature branch.
- Team members will review the code using GitHub's PR review user interface.
  - ***We'll provide a checklist and document to record your suggested changes***
  - ***Team submits the document to a Dropbox***
- After the changes are approved, the feature branch is merged into master

Issuing pull requests and performing code reviews will now be a part of your development workflow.



Definition of Done Checklist

Delete...

0%

- [ ] acceptance criteria are defined
- [ ] solution tasks are specified
- [ ] feature branch created
- [ ] unit tests written
- [ ] solution is *code complete*, i.e. passes full suite of unit tests
- [ ] design documentation updated
- [ ] pull request created
- [ ] user story passes all acceptance criteria
- [ ] code review performed
- [ ] feature branch merged into master
- [ ] feature branch deleted

- The *Pull Request* is made when the story moves to *Ready for Test*, i.e. after the user story is code complete, and the design documentation is updated.
- Review should be done by a minimum of two team members other than the developer of the story.
- Acceptance testing can be performed in parallel.