CSED332: Software Design Methods

Lecture 1: Software Process & Configuration Management

Kyungmin Bae

Department of Computer Science and Engineering
POSTECH

Fall 2018

Course Staff

- Instructor: 배경민 Kyungmin Bae
    - Office: B2 225
    - Email: kmbae@postech.ac.kr
- Teaching assistants
    - 김사론 Saron Kim (ksron@postech.ac.kr)
    - 이지아 Jia Lee (cee5539@postech.ac.kr)

Prerequisites

- Required
  - data structure
  - object-oriented programming
  - some programming skills
- Not required: Java
  - programming language used in the class
  - very similar to C++
  - small assignments to help you learn Java

Course Materials

- Recommended textbooks (not required)
  - Ian Sommerville, "Software Engineering" (9th or 10th)
  - Steve McConnell, "Code Complete" (2nd)
- More resources listed on LMS
  - `http://lms.postech.ac.kr`
  - lecture notes, book chapters, papers, etc.
- Book chapters and papers
  - reading will be on exams
  - first reading will be posted today

Grading

- Homework (20%)
  - learn tools and practices
- Team project (35%)
  - work in larger groups (size 8)
- Exams (40%)
  - midterm and final exams
- Attendance (5%)
  - No execute unless informed in advance
  - 8 or more absences mean F

Academic Integrity

- Violations of academic integrity will severely affect your grade.
- Messing with git history is considered CHEATING

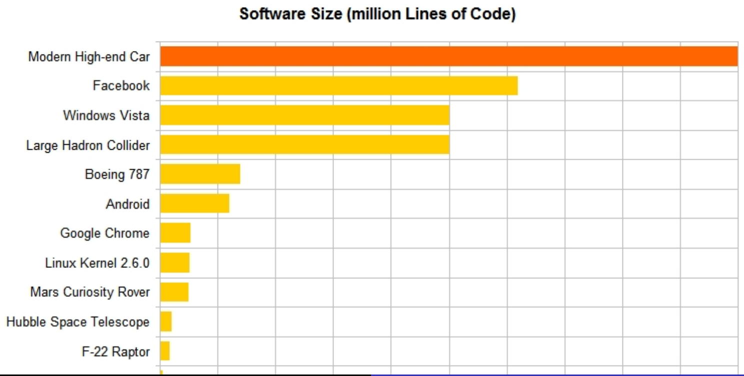
Course Overview

Challenges in Software Design

- Growth of code and complexity over time

1993	Windows NT 3.1: 4–5 million
1996	Windows NT 4.0: 7–8 million
2000	Windows 2000: 29 million
2003	Windows Server 2003: 50 million

[Knowing.NET. 2005]



Challenges in Software Design

- Software has bugs

Ariane 5 Rocket Explosion, 1996

Toyota's ETCS bugs, 2009–11

North America blackout, 2003

How to Design Software?

Stone Age Software Designs

- Writing algorithms/data structures from scratch?
  1. discuss software that need to be written
  2. write some code
  3. test the code, and fix the defects
  4. if not done, return to step 1
- Hard to
  - design, implement, ...
  - test, verify, ...
  - maintain, change, ...
  - collaborate, finish on time, ...

How to Design Software?

Better Software Design

- Metrics of software quality
  - functional correctness
  - reliability
  - flexibility
  - security
  - ...
- Use a design process
  - think and analyze before coding
  - consider non-functional quality attributes
  - explicitly consider constraints and costs
  - facilitates communication
  - improves software quality

What is (Not) Software Engineering?

- Not just software programming
  - individual vs. team
  - program vs. software
- Not just a process
  - field that studies several different processes and techniques
  - include mathematical, algorithmic, or automated methods

Some Definitions of Software Engineering

- IEEE 610

  The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software.

- Bauer

  The establishment and use of sound engineering principles in order to economically obtain software that is reliable and works efficiently on real machines.

- Sommerville

  An engineering discipline that is concerned with all aspects of software production

Software Varies

- Size
- How humans interact with it
- Requirements stability
- Need for reliability
- Need for security
- Portability
- Cost

Software Varies

Example: Office Software

Size	large
Interactiveness	high
Requirements	frequent new features
Reliability	moderate
Security	low (at least used to be)
Portability	high
Cost	high

Software Varies

Example: Space Shuttle Software

Size	moderate to large
Interactiveness	low
Requirements	stable
Reliability	very high
Security	low
Portability	low
Cost	high

Software Varies

Example: Online Shopping Software

Size	moderate
Interactiveness	high
Requirements	frequent new features
Reliability	high
Security	high
Portability	low
Cost	low

Software Varies

Your Example

- Size:
- Interactiveness:
- Requirements:
- Reliability:
- Security:
- Portability:
- Cost:

Many Way to Develop Software

- Plan-driven / agile
- Centralized / distributed
- High math / low math
- Close / little interaction with customers
- Much testing / little testing
- Organize by architecture / features

Software Process

- Pressman

  A framework for the tasks that are required to build high-quality software

- IEEE 1074

  A set of activities performed towards a specific purpose

- Sommerville

  a structured set of activities required to develop a software system

Activities in IEEE 1074

- Project Management
  - project initiation/planning/control, software quality management
- Development
  - requirements, design, implementation
- Post-development
  - installation, operation, maintenance, retirement
- Integral processes
  - verification and validation, software configuration management

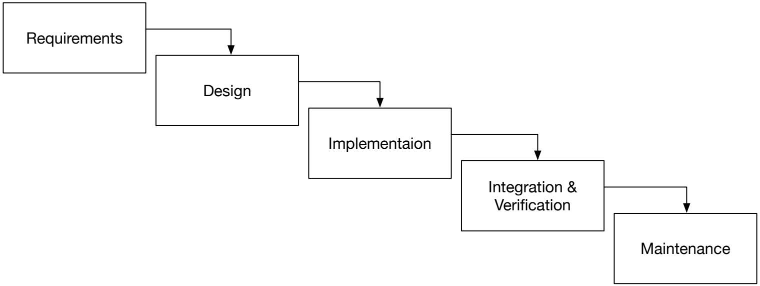
Defined Processes

- Agile
  - eXtreme Programming (XP), Scrum, ...
- Formal
  - Rational Unified Process, Cleanroom, ...
- Open-source
  - Bazaar, Cathedral, ...
- ...

Software Process Models

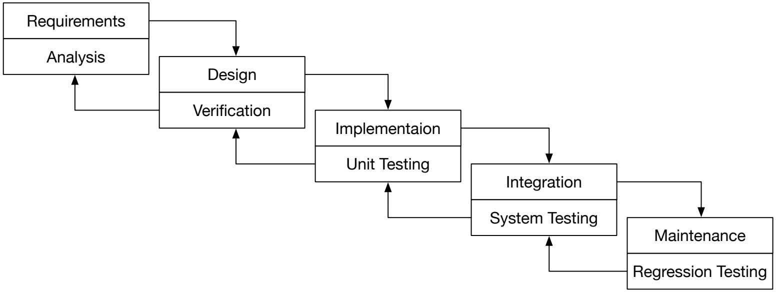
- Simplified representation of software process
  - particular perspective of software process
  - explain different software development approaches
- Examples
  - waterfall, incremental model, prototyping, spiral model, . . .
- Modern processes involve several process models (in part)

Software Process Models: Waterfall



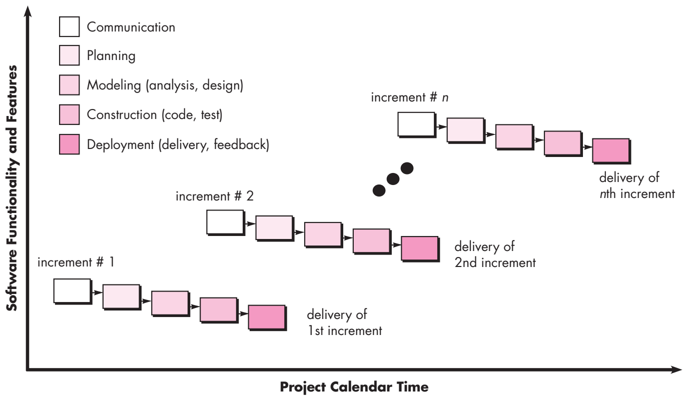
- first process model
- plan-driven and sequential
- difficult to accommodate changes

Software Process Models: Modified Waterfall



- waterfall + quality assurance
- plan-driven and sequential
- difficult to accommodate changes

Software Process Models: Incremental Development



Communication
Planning
Modeling (analysis, design)
Construction (code, test)
Deployment (delivery, feedback)
increment # 1
delivery of 1st increment
increment # 2
delivery of 2nd increment
increment # *n*
delivery of *n*th increment
Software Functionality and Features
Project Calendar Time

- combines linear and parallel process flows
- fundamental part of agile approaches
- easier to get customer feedback

Activities: Requirements

- What software should do
  - interfaces, functionality, constraints, . . .
- Requirements analysis
  - remove ambiguities, inconsistencies, and incompleteness

Activities: Design

- How software should do
    - define modules, and write their specifications
- Verification
    - conformance to requirements

Activities: Implementation and Integration

- How modules work and are integrated
  - executable code, data base, documentation, . . .
- Unit testing
  - each module in isolation
- System testing
  - module interactions

Activities: Maintenance

- Evolve software
  - as requirements change, new technologies, new platforms, ...
- Regression testing
  - test changes

Activities: Validation

- Does software meet requirement?
  - requirements, design, specifications, . . .
- Related to every development activity
  - requirement/design validation, unit/system/regression testing, . . .

Purpose of Course

- Learn a particular process (XP)
  - how to follow a process
  - how to change/improve a process
- Learn steps common to most processes
  - software configuration management (SCM), testing, metrics, documentation, refactoring, reverse engineering, debugging
- Learn techniques for software design
  - design patterns, framework, software architecture
  - unified modeling language, software verification

Project

- Learn architecture of a relatively large software project
  - Eclipse (or IntelliJ) plugins
- Learn modern SE practices
  - SCM, unit testing, automated build, documentation, refactoring
- Learn modern SE environment
  - Eclipse (or IntelliJ)
- Teamwork to have realistic experience (up to 8 students)
  - participation is mandatory (will be graded)

Software Configuration Management

Software Product

- Code
- Test suites
- Operation manuals
- Requirements and specifications
- Design documentation
- Plans and schedules
- . . .

Need to keep track of how you created software!

Software Configuration Management (SCM)

- Definition (by the Software Engineering Institute)
  - discipline for controlling the evolution of software systems
- Four major aspects (among many others)
  - version control, building, change management, releasing
- More important on large projects
  - requires expertise and oversight, and often supported by tools

How to Keep Track of Changes?

- Good old way

myFile-1.txt myFile-2.txt myFile-3.txt myFile-4.txt

- When / what did I change?

myFile-aug4morning.txt myFile-aug4later-in-the-morning.txt myFile-aug4-evening.txt myFile-aug9.txt

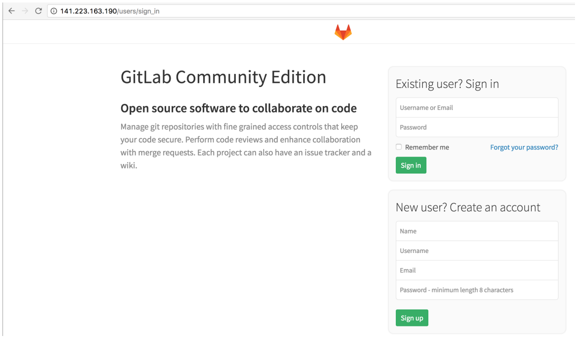
myFile-final.txt myFile-latest.txt myFile-last.txt myFile-final-final.txt

- Version control system (VCS) to the rescue!

Version Control System (VCS)

- Keep track of the multiple versions of system components
  - collaborate on a project with multiple other developers
  - revert changes, or go back in time to a specific version
- Examples
  - Git, SVN, CVS, . . .
  - we will use Git in CSED332

Git Remote Repository (1)


141.223.163.190/users/sign_in

GitLab Community Edition

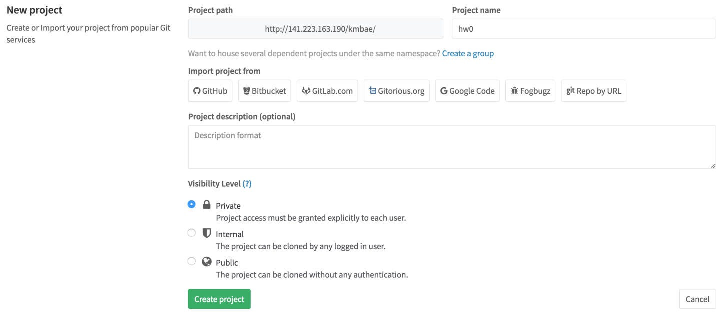
Open source software to collaborate on code

Manage git repositories with fine grained access controls that keep your code secure. Perform code reviews and enhance collaboration with merge requests. Each project can also have an issue tracker and a wiki.

- We will use http://141.223.163.200 for this class
- Everyone should create an account (use your Hemos ID).
- Create SSH keys (see http://141.223.163.200/help/ssh/README)

Git Remote Repository (2)

- Submit homework and project assignments to the server.
- Create a private project for individual assignments.



New project

Create or Import your project from popular Git services

Project path	Project name
http://141.223.163.190/kmbae/	hw0

Want to house several dependent projects under the same namespace? Create a group

Import project from

Project description (optional)

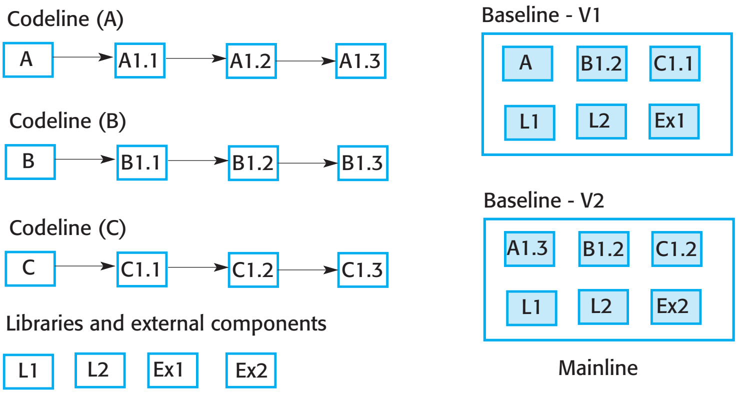
Description format

Visibility Level (?)

- [✓ Private] Project access must be granted explicitly to each user.
- Internal: The project can be cloned by any logged in user.
- Public: The project can be cloned without any authentication.

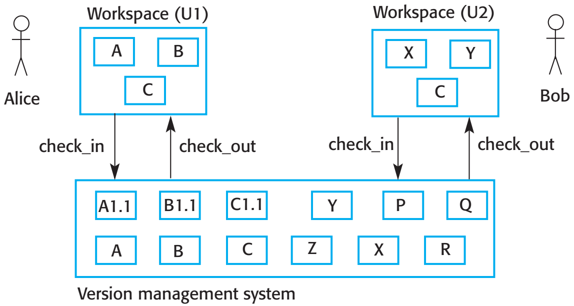
- Team leader will invite team members for project assignments.

Codeline, Baseline, and Release



- Codeline: a sequence of versions of source code
- Baseline: a set of component versions that make up a system
- Release: a baseline that the developers give to other people

Check-in and Check-out



- Repository contains complete history (local or remote)

Version Control Workflow

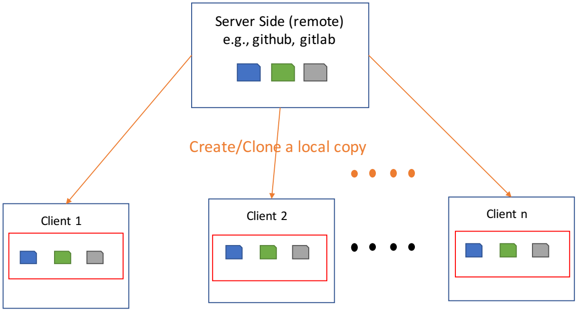
1. Repository contains complete history
2. Pull latest version
3. Work on the code
4. Commit and then push, making a new version of the software

Version Control: Parallel Work

- What happens if two people change same software at same time?
  - A pulls V23, changes it, and pushes V24
  - B also pulls V23, changes it, and pushes ??
- Approach 1: locking (impossible)
  - first one locks software, and second one waits until lock is released
  - very inefficient and essentially sequential
- Approach 2: merging (common)
  - second person merges the changes (mostly automatically)
  - sometimes can introduce bugs (testing required)

Git Basics

Cloning a Local Copy of Remote Repository



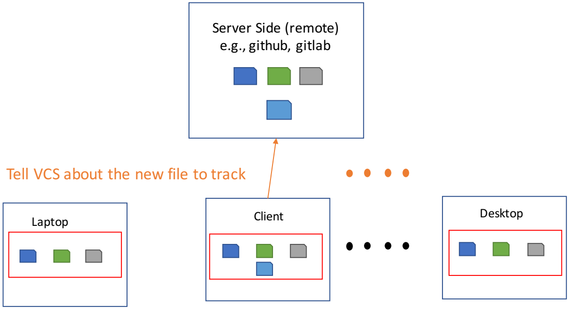
- Create a local copy of the remote server:

  `git clone git@141.223.163.200:`⟨HemosID⟩`/`⟨ProjectName⟩`.git`

- A local copy named ⟨ProjectName⟩ is created
  - that replicates the remote repository on gitlab

Git Basics

Adding a New File to Track



- Track a new file or stage a modified file

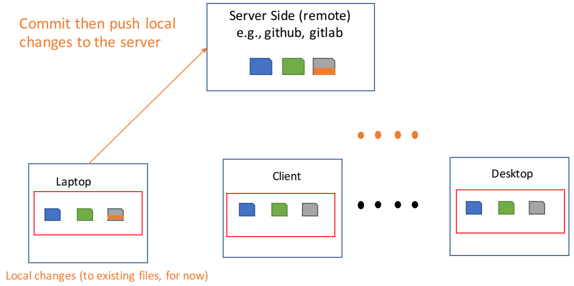
  `git add` ⟨FileName⟩

- Check the status of your (tracked) files

  `git status`

Git Basics

Commit/Push Local Changes



Commit then push local changes to the server
Server Side (remote) e.g., github, gitlab
Laptop
Client
Desktop
Local changes (to existing files, for now)

- Commit (tracked) local changes made to your local repository

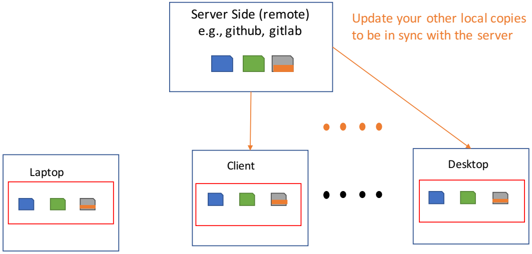
  `git commit -m '⟨a meaningful descriptive message⟩'`

- Push your commits to the remote repository

  `git push origin master`

Git Basics

Updating Other Local Copies


Server Side (remote)
e.g., github, gitlab
Update your other local copies to be in sync with the server
Laptop
Client
Desktop

- Pull changes from the remote repository

```
git pull origin master
```

- View the commit history

```
git log
```

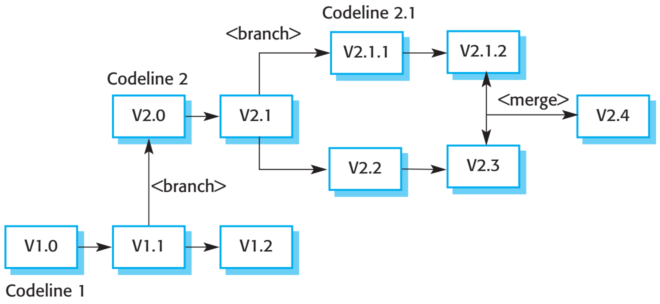
Git References

- Git help command

  ```
  git help
  ```

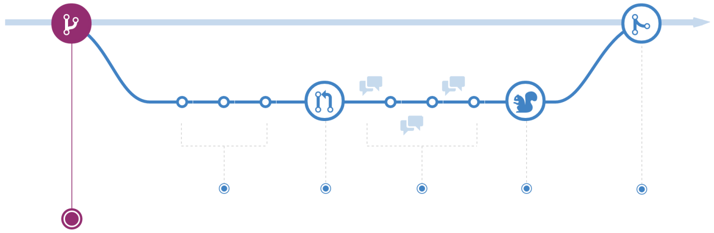
- Tutorial and Documentation:
  - `https://try.github.io/`
  - `https://www.atlassian.com/git/tutorials/`
  - `https://git-scm.com/doc`
- Git in Eclipse
  - `http://eclipsesource.com/blogs/tutorials/egit-tutorial`

Branching



- Encourage (use of short-lived) branches
  - new functions, fixing bugs, experimental versions, ...
- Bad reasons to branch
  - support different hardware platform, different customer, ...

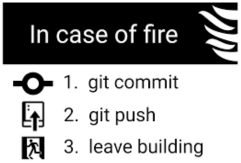
Branching and Merging in Git[1]



- Create a branch $\longrightarrow$ Add commits $\longrightarrow$ Merge (or pull) request $\longrightarrow$ Code review $\longrightarrow$ Merge

Golden Rule of VCS

- Commit (and push) frequently!
  - break work into small steps
  - commit the changes for each step, and push it on the same day
- The longer you have code uncommitted/unpushed
  - the more you interfere with others
  - the harder it is to merge later

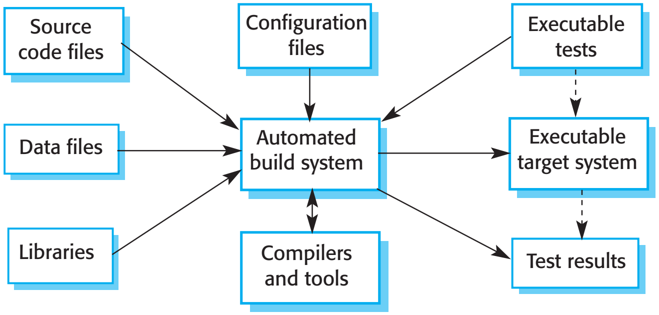

In case of fire

1. git commit
2. git push
3. leave building

Build Management

- How do you build the product?
  - which compiler, flags, source files, libraries, . . .
  - which versions?
- Building should be automatic
  - build regularly to test build procedure
  - should minimize redundant recompilation
- Need a tool
  - make, ant, mvn, gradle, MSBuild, . . .

Build System



Maven Snippet

```
<project xmlns="http://maven.apache.org/POM/4.0.0" ... >
  <modelVersion>4.0.0</modelVersion>
  <groupId>iTrust</groupId>
  <artifactId>iTrust</artifactId>
  <version>21.0.01</version>
...
  <build>
    <sourceDirectory>src</sourceDirectory>
    <testSourceDirectory>test</testSourceDirectory>
...
  </build>
  <dependencies>
      <dependency>
          <groupId>org.apache.tomcat</groupId>
          <artifactId>tomcat</artifactId>
          <version>8.0.28</version>
          <type>pom</type>
      </dependency>
...
  </dependencies>
...
</project>
```

Maven: Build the Project

```
$ mvn package
 ...
[INFO] ----------------------------------------
[INFO] BUILD SUCCESSFUL
[INFO] ----------------------------------------
[INFO] Total time: 2 seconds
[INFO] Finished at: ...
[INFO] Final Memory: 3M/6M
[INFO] ----------------------------------------
```

- Maven phases
  - validate, compile, test, package, install, deploy, clean, . . .

Maven References

- Maven help command

```
mvn -help
```

- Tutorial
  - `https://maven.apache.org/guides/getting-started`
- Maven in Eclipse
  - `http://www.eclipse.org/m2e/`

Daily Build and Smoke Testing

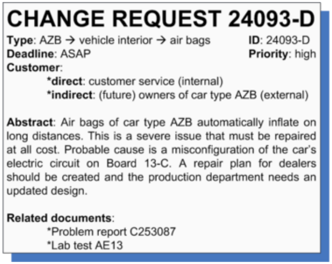
- Build process can be broken
  - check in bad code
  - forget to include file in build scripts
  - move a library
- Every day or night
  - build the latest version of product
  - run simple test suite (called smoke testing)

Change Management

- Manage changes
  - keep track of how product changes over time
  - be able to reproduce any version of it
- Control how product changes
  - needed changes have been made
  - no improper changes have been made
- Not only code, but also non-code resources: e.g.,
  - test suite for version 6.4.3 does not work for other versions.
  - manual for version 6.4.3 is slightly different than for other versions

Change Request

- We decided to implement a change
  - Has it been implemented fully? (tests, code, manuals, ...)
  - What parts of the system were affected by that change?
- I look at a program or document
  - Why is it like this?
  - When was it written, by whom?
- Tracing both ways
  - change management
  - version control linked through IDs



CHANGE REQUEST 24093-D

Type: AZB → vehicle interior → air bags **ID**: 24093-D
Deadline: ASAP **Priority**: high
Customer:
*direct: customer service (internal)
*indirect: (future) owners of car type AZB (external)

Abstract: Air bags of car type AZB automatically inflate on long distances. This is a severe issue that must be repaired at all cost. Probable cause is a misconfiguration of the car's electric circuit on Board 13-C. A repair plan for dealers should be created and the production department needs an updated design.

Related documents:
*Problem report C253087
*Lab test AE13

Activities in Change Management

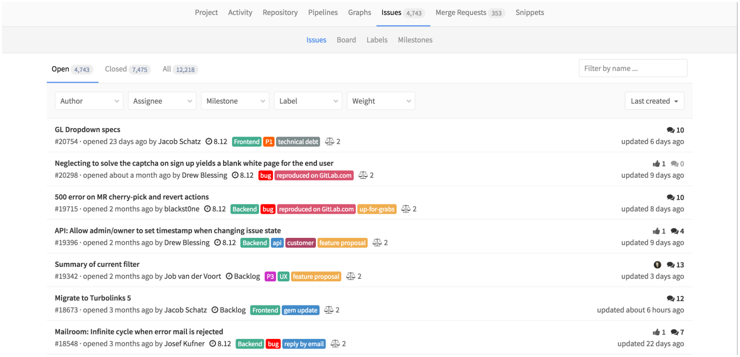
- Change request or order
  - new feature or bug report
- Change control authority
  - decides which changes should be carried out
- Should link code changes to change requests
  - which part and version

Example: Bugzilla

- Originally developed for Mozilla
  - `http://bugzilla.mozilla.org`
  - `https://developer.mozilla.org/en/Bug_writing_guidelines`
- Database for bugs
  - lets people report bugs
  - assigns these bugs to the appropriate developers
- Each bug report
  - ID, name of reporter, description, component, . . .
  - status: unconfirmed, new, resolved, verified, closed

Example: Issue Tracking

- Provided by many repository hosting services
  - GitHub, GitLab (this class), SourceForge, Bitbucket, . . .
- Example: GitLab
  - each issue has ID, reporter, description, developer, related branch



Project Activity Repository Pipelines Graphs Issues 4,743 Merge Requests 353 Snippets
Issues Board Labels Milestones
Open 4,743 Closed 7,475 All 12,218
Filter by name ...
Author Assignee Milestone Label Weight
Last created

GL Dropdown specs
#20754 · opened 23 days ago by Jacob Schatz 8.12 Frontend P1 technical debt 2 — 10 — updated 6 days ago

Neglecting to solve the captcha on sign up yields a blank white page for the end user
#20298 · opened about a month ago by Drew Blessing 8.12 bug reproduced on GitLab.com 2 — 1 0 — updated 9 days ago

500 error on MR cherry-pick and revert actions
#19715 · opened 2 months ago by blackst0ne 8.12 Backend bug reproduced on GitLab.com up-for-grabs 2 — 10 — updated 8 days ago

API: Allow admin/owner to set timestamp when changing issue state
#19396 · opened 2 months ago by Drew Blessing 8.12 Backend api customer feature proposal 2 — 1 4 — updated 9 days ago

Summary of current filter
#19342 · opened 2 months ago by Job van der Voort Backlog P3 UX feature proposal 2 — 13 — updated 3 days ago

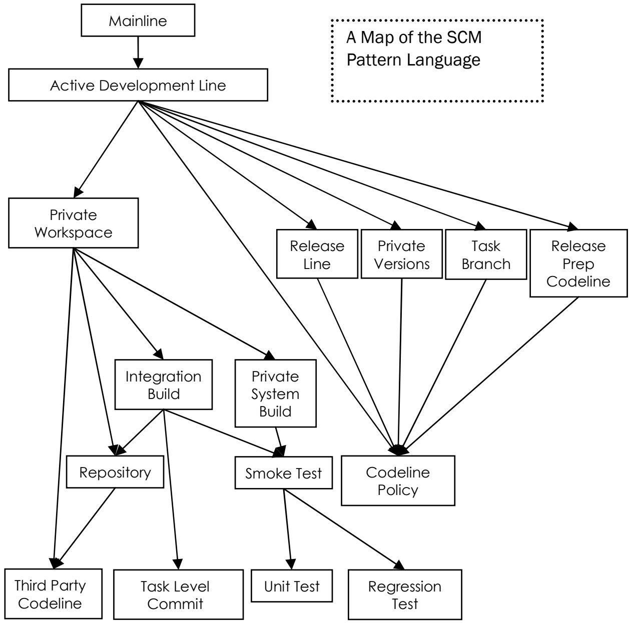
Migrate to Turbolinks 5
#18673 · opened 3 months ago by Jacob Schatz Backlog Frontend gem update 2 — 12 — updated about 6 hours ago

Mailroom: Infinite cycle when error mail is rejected
#18548 · opened 3 months ago by Josef Kufner 8.12 Backend bug reply by email 2 — 1 7 — updated 22 days ago

SCM in Practice

- SCM Manager
  - maintain tools
  - maintain configuration files, make branches
  - do the merging
  - create policies on version control and change control
- Alternatives
  - toolsmith supports SCM tools
  - architect defines configuration files
  - developers merge their code back into mainline
  - managers define policies for version control and change control

Software Configuration Management Patterns



Various Tests

- Smoke test
  - ensure that the system still runs after you make a change
- Unit test
  - ensure that a module is not broken after you make a change
- Regression test
  - ensure that existing code does not get worse

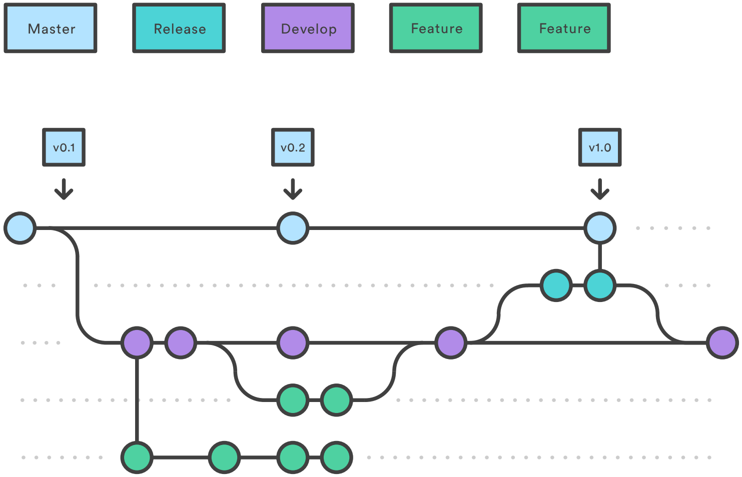
Developer Issues

- Private workspace
  - prevent integration issues from your changes
- Private system build
  - avoid breaking build before committing changes to the repository
- Many other “development rules”
  - typically enforced by companies

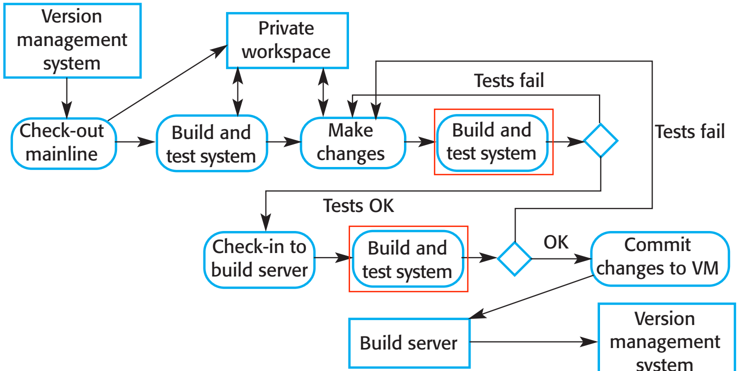
Codeline Policy

- Active Development Line
  - keep a rapidly evolving codeline stable enough to be useful
- Release Line
  - Holds bug fixes for a release
- Private Versions
  - experiment with complex changes locally
- Task Branch
  - Hide a disruptive task from the rest of the team
- Release Prep Codeline
  - stabilize a codeline for an upcoming release

Git Workflow



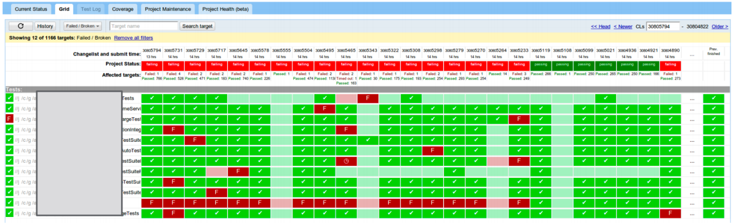
Continuous Integration (CI)



- Building and testing software projects continuously
- Need automated building and testing

CI at Google

- Google's Test Automation Platform (TAP)



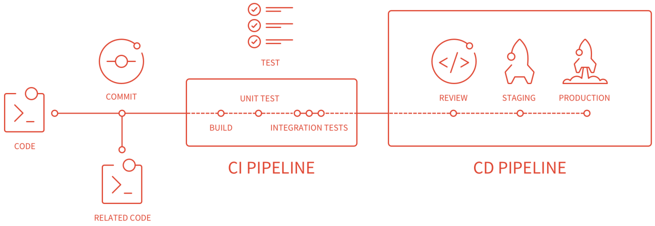
Showing 12 of 1166 targets: Failed / Broken Remove all filters

	Changelist and submit time:	30805794 13 hrs	30805731 14 hrs	30805729 14 hrs	30805717 14 hrs	30805645 14 hrs	30805578 14 hrs	30805555 14 hrs	30805504 14 hrs	30805495 14 hrs	30805465 14 hrs	30805343 14 hrs	30805322 14 hrs	30805308 14 hrs	30805298 14 hrs	30805279 14 hrs	30805270 14 hrs	30805264 14 hrs	30805233 14 hrs	30805119 14 hrs	30805108 14 hrs	30805099 14 hrs	30805021 14 hrs	30804936 14 hrs	30804921 14 hrs	30804890 14 hrs	...	Prev. finished
	Project Status:	failing	failing	failing	failing	failing	failing	failing	failing	failing	failing	failing	failing	failing	failing	failing	failing	failing	failing	passing	passing	passing	passing	passing	passing	failing		
	Affected targets:	Failed: 1 Passed: 766	Failed: 4 Passed: 528	Failed: 2 Passed: 471	Failed: 2 Passed: 183	Failed: 2 Passed: 740	Failed: 1 Passed: 226	Passed: 1	Failed: 1 Passed: 474	Failed: 2 Passed: 113	Failed: 2 Timed out: 1 Passed: 163	Failed: 1 Passed: 30	Failed: 1 Passed: 175	Failed: 1 Passed: 193	Failed: 2 Passed: 254	Failed: 1 Passed: 293	Failed: 1 Passed: 254	Passed: 14	Failed: 3 Passed: 249	Passed: 266	Passed: 1	Passed: 250	Passed: 265	Passed: 250	Passed: 166	Failed: 1 Passed: 273		
Tests:																												
✓	//j /c/g /a [redacted] Tests	✓	✓	✓	✓					✓		F		✓									✓				...	✓
✓	//j /c/g /a [redacted] meServi	✓	✓	✓	✓	✓	✓		✓	F	✓		✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	...	✓
F	//j /c/g /a [redacted] argeTest	✓	✓	✓	✓	✓	✓		✓	✓	✓		✓	✓	✓	✓	✓	✓	F	✓		✓	✓	✓	✓	✓	...	✓
✓	//j /c/g /a [redacted] ionInteg	✓	F	✓	✓	✓	✓		✓	✓	F		✓	✓	✓	✓	✓		✓	✓		✓	✓	✓	✓	✓	...	✓
✓	//j /c/g /a [redacted] TestSuite	✓	✓	F	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓		✓	✓	✓	✓	✓	...	✓
✓	//j /c/g /a [redacted] AutoTest	✓	✓	✓	✓	✓	✓		✓	✓	✓		✓	✓	F	✓	✓		✓	✓		✓	✓	✓	✓	✓	...	✓
✓	//j /c/g /a [redacted] estSuite	✓	✓	✓	✓	✓	✓		✓	✓	◷		✓	✓	✓	✓	✓		F	✓		✓	✓	✓	✓	✓	...	✓
✓	//j /c/g /a [redacted] estSuiteF	✓	✓	✓		F	✓		✓	✓						✓				✓						✓	...	✓
✓	//j /c/g /a [redacted] TestSui	✓	F	✓	✓	✓	✓		✓	✓	✓		✓	✓	✓	✓	✓		✓	✓		✓	✓	✓	✓	✓	...	✓
✓	//j /c/g /a [redacted] estSuite	✓	✓	✓	F	✓	✓		✓	✓	✓		✓	✓	✓	✓	✓		✓	✓		✓	✓	✓	✓	✓	...	✓
✓	//j /c/g /a [redacted]	F	F	F	F	F	F		F	F	F		F	F	F	F	F		F	✓		✓	✓	✓	✓	✓	...	✓
✓	//j /c/g /a [redacted] eTests	✓	F	✓	✓	✓	✓		✓	✓	✓		✓	✓	✓	✓	✓		✓	✓		✓	✓	✓	✓	F	...	✓

- Managing 2 billion lines of code

CI in GitLab

- `https://docs.gitlab.com/ee/ci/`



- Configured by the file `.gitlab-ci.yml` in the repository.
  - defines a set of jobs to be executed

Summary

- Software process
  - a set of activities required to develop a software system
- Software configuration management
  - version control, building, change management, releasing
- Homework 1 (due 9/13)
  - Create your Git account on: `http://141.223.163.200`
  - Try out Eclipse and Java
- Reading
  - `https://en.wikipedia.org/wiki/Software_development_process`
  - `http://www.scmpatterns.com/book/SCMPatterns-RefCard.pdf`

Questions?