Översikt över vanliga metoder & verktyg

Uppgift 1 – Metoder & verktyg för mjukvara(DT042G)

David Hegardt Martin Degerman David Jonsson

March 31, 2016

Contents

1 Stödjande

GIT

GIT är ett verktyg för versionshantering utvecklat utav skaparen av Linux, Linus Thorvalds. Versionshantering används inom projekt för att kunna spåra ändringar som gjorts i filerna som hör till projektet, oftast används versionshantering för skriven kod i systemutveckling. Versionskontroll är viktigt att använda för att undvika att skriva över kod av misstag, men det är absolut nödvändigt i större projekt där man arbetar med flera personer. Med hjälp av en mjukvara för versionshantering sköts detta av ett program istället för att manuellt spara ner versioner. Med mjukvara för versionshantering kan man sedan gå tillbaka och se tidigare gjorda ändringar i koden.

GIT är ett distribuerat system för versionshantering. Detta innebär att systemet inte bara hanterar versionshantering lokalt, eller på en server utan flera kopior av repositoryt existerar på flera olika maskiner. På detta sätt bygger man ett redundant system, om filerna försvinner från servern eller från en dator så finns det fortfarande kvar flera kopior hos medlemmarna samt på servern där projektet är uppladdat. Med hjälp av GIT kan också flera versioner av projektet finnas lagrat på olika klienter. Fördelen är också att medlemmarna som arbetar i projektet inte behöver vara uppkopplade mot servern när de jobbar mot projektet, det behöver bara göras när man hämtar data från repositoryt (pull) eller när man gjort ändringar som man vill ladda upp (push).
GIT skiljer sig från andra mjukvaror för versionshanteringen, andra system tar hänsyn till vilka filer som har ändrats för att hantera olika versioner av dessa. GIT använder sig istället av sk snapshots av hela filstrukturen, en snapshot är en bild av hela filsystemet vid ett visst tillfälle. En snapshot skapas när man gör en commit, om ingenting har förändrats så skapas en länk till den tidigare versionen, om något har förändrats så sparas den förändrade information i den nya snapshot kopian.

Github

Github är en hemsida med serverlagring av GIT projekt. Det är fullt möjligt att använda GIT utan att använda Github, men man behöver då själv sätta upp en server. Github bygger på open source principen med att det mesta laddas upp för allmän beskådan, vem som helst kan komma åt och se källkoden för de projekt som är uppladdade på Github. Andra git användare kan också vara med och vidareutveckla projektet, dessa blir då projektmedlemmar som kallas för collaborators. En användare kan begära en pull request på ett projekt, och ägaren av projektet kan sedan gå in och bevilja detta via hemsidan vilket tillåter användaren att ladda ner hela projektet till sin klient. När man

använder sig av GIThub så får man en GUI för att överblicka projektet via hemsidan, härifrån kan man se samtliga versioner av projektets filer samt vilka ändringar som gjorts mellan olika versioner. Man kan se vem som gjort vilken ändring och vad som skiljer de olika versionerna åt.

2 Testverktyg

Google test

Google test är ett ramverk för enhetstestning i C++. Enhetstestning är väldigt enkelt, och går i stort sett ut på att berätta vad en funktion förväntas göra, och se om den också gör det.

När man väl kommit igenom installationen och fått ordning på de grundläggande inställningarna, är det ungefär lika svårt att köra sina tester som att kompilera och köra sitt program. Resultatet av testerna är nästan omöjligt att missförstå, och visas direkt i terminalen.

Det enda som egentligen riskerar ställa till det för en nybörjare är just de första inställningarna. Den som vill använda ramverket hänvisas till dokumentationen för sitt buildsystem, och endast den som har tillräcklig tur kommer snubbla över en forumtråd eller annan obskyr informationskälla med information om möjligheten att använda en linkerflagga för att bygga testprogrammet.

Hur själva testklasserna och testerna utformas är däremot mycket väldokumenterat. Med början i ett primerdokument om grunderna kan även en fullständig nybörjare snabbt komma in i arbetet, och också gå vidare till mer avancerade guider.

Gcov

GNU/gcc tillhandahåller verktyget gcov för code coverage-testning. Även här är den grundläggande användningen väldigt enkel, och går i stort sett ut på att man anropar gcov med källkoden som ska testas, för att sedan köra resulterande fil. Resultatet presenteras i defaultläget som en sammanställning av antal kodrader, antal lästa, och lästa i procen, men flera flaggor kan användas för att få fram mer utförlig information. Code coverage blir som mest värdefullt när det kombineras med enhetstestning, och förmågan att se inte bara hur testad kod klarar sig, utan också hur stor del av koden som testerna faktiskt går igenom kan bidra till både säkrare och mer optimerad kod.

Nackdelen med både enhetstestning och code coverage är förstås att programmering är en för komplex syssla för att så enkla kvantitativa mätmetoder ska kunna ge entydiga resultat, mer än i de allra enklaste fallen. Ingenting säger att de tester som *inte* skrivits inte heller hade behövts. Och inga garantier finns för att de tester som skrivits är meningsfulla, hur stor del av koden som än omfattas. Risken finns till och med, och då särskilt för nya programmerare, att man lockas att jaga procent, istället för att ta fram så relevanta tester som möjligt.

Referenser

Chacon, Scott, and Ben Straub. 2016. “Pro GIT - 2nd Edition.” March 3. https://git-scm.com/book/en/v2.

Donahue, Billy. 2016. “Goggletest - Primer.” February 16. https://github.com/google/googletest/commits/master/googletest/docs/Primer.md.

unknown. 2016. “Gcov - a Test Coverage Program.” March 15. https://gcc.gnu.org/onlinedocs/gcc/Gcov.html#Gcov.