Om bugs i kode

Dag Sverre Seljebotn (gruppelærer INF3/4130 h07)

18. oktober 2007

I forbindelse med oblig 1 var det en del som ga tilbakemelding om at ting tok mye tid fordi de satt i timer eller dager for å finne små bugs. Det inspirerte meg til å skrive et lite notat som egentlig er helt på siden av INF3/4130, men som likevel kan være relevant etter oblig 1. Det er jo to obliger igjen, så... selvfølgelig hører dette helst innunder helt andre kurs, men siden det delvis var et problem på oblig 1 i dette kurset så er jeg freidig nok til å streife innom det.

Som vanlig er alle meninger mine egne. Og her er det også en fare for at jeg sier ting som en vil være veldig uenig i – gi i såfall tilbakemelding så hiver vi inn diskusjonen i notatet. (Ellers vil det at en har meninger i det hele tatt på dette feltet være et veldig godt tegn, for da har en tenkt gjennom saken og det er det som er poenget).

(Jeg er helt med på at det kanskje ikke er så mye vits i et slikt notat siden programmering stort sett er en erfaringssak – på den annen side, hvem er det som begynner med TDD (se lenger ned) helt av seg selv?)

1. Start helt på nytt om en bug tar mer enn en time å finne. En sitter og tenker "om jeg bare ser fem minutter til på koden så finner jeg den", men om en har den tanken i flere timer sammenhengende så må en heller stoppe opp. En gjennomsnittlig oblig-oppgave burde ta en gjennomsnittlig programmerer mellom tjue minutt og et par timer å kode om en kjenner algoritmen godt, så om en bruker fire timer på å lete etter en bug kunne en faktisk skrivd programmet sitt på nytt flere ganger i stedet – og antagelig eliminert bugen en av de gangene. Om en ikke gjør det kan en være rimelig sikker på at en ikke forstår algoritmen godt nok – og da er det penn og papir og en stille lesesal som gjelder!

2. Bugs er ikke en del av hverdagen – en skal unngå å lage de i det hele tatt. Mange har en utviklingsmetodikk som går ut på å a) hacke sammen

noe, b) finn bugsa, c) gå til a. Men om en har mange bugs bør en kanskje venne seg til andre metodikker også. Det er mange eksempler på at små isolerte oppgaver av typen en får i dette kurset kan skrives ned uten å få en eneste bug på første forsøk. Om bugs er et problem så sakk ned og prøv å gjøre det på den måten.

3. Skriv ren og pen kode – for renhetens og penhetens egen del. Programmerere kan ofte plasseres på en skala der den ene ekstremiteten bare bryr seg om at programmet fungerer riktig og den andre ekstremiteten bare bryr seg om at koden ser ryddig, lekker og flott ut. Sistnevnte kan høres kjedelig ut, men gjett hvem som gjennomsnittlig skriver færrest bugs i utgangspunktet? Tenk over hvor du vil plassere deg på en slik skala.

4. Les andres kode Vær ikke bare opptatt av å skrive egen kode, men også av å lese andres kode.

5. Metodologier Tenk gjennom om det finnes anerkjente utviklingsmetodologier en kan lære noe av. Knuth selv (eller var det noen andre?) var tilhenger av å bevise matematisk at hver eneste kodelinje hadde et formål og var riktig. En helt annen tankegang (som nok er mer i vinden i IT-miljøer nå) er TDD. Les denne:
`http://en.wikipedia.org/wiki/Test_driven_development`
Det finnes flere metodologier også, poenget må være å kjenne til noen og tenke på om en kan integrere deler av det på en praktisk måte.