Fakulteten för ekonomi, kommunikation och IT

Resultat av Mätprogram

Orfeus

Datum: 2006-12-26
Namn: Henrik Bäck
Mathias Andersson
Kurs: DAV C22

Innehållsförteckning

Inledning

Den här rapporten redovisar resultatet av det mätprogram som satts upp för projektet Orfeus. Orfeus utvecklas under hösten 2006 vid Karlstads Universitet.

I den här rapporten kommer de frågeställningar som ställts att utredas och besvaras efter bästa förmåga. Vissa frågor har inget svar på grund av projektets natur och på det sättet projektet just nu drivs.

Programkodsunderlaget för denna rapport baseras på version 1.27 från den 14 december 2006. Samma version av koden kommer att användas genom hela rapporten trots att nya versioner kommer att släppas under tiden rapporten skrivs.

Återkommande definitioner

Den här rapporten kommer återkommande gånger använda ett antal begrepp. För att bringa klarhet i dessa begrepp kommer de att definieras i denna sektion. Övriga begrepp förklaras under respektive sektion.

Programkodsrad (NCLOC)

Orfeus skrivs i C-kod och med avseende på detta har definitionen av programkodsrad gjorts. I den här rapporten kommer en (1) programkodsrad att räknas som en (1) rad vilken innehåller ett programpåstående och som har avslutats med ett semikolon. Till detta räknas även slingor, loopar och val men även inkluderingar av bibliotek och definitioner. Alltså räknas bland annat inte tomma rader eller rader med kommentarer till denna mängd.

Projektets mognadsgrad

Ett projekts mognadsgrad kan bestämmas med metoden CMM. Alla projekt befinner sig åtminstone i nivå ett (1) av de fem (5) nivåer som finns. Projektgruppen utvecklar enligt XP-metoden och den har i sin natur svårt att nå upp till nivå två (2) av CMM. Därför har några av de krav som ställs på ett projekt för nivå två (2) i CMM valts ut för en enkät.

Efter ett första möte med projektgruppen fanns vissa misstankar om att alla i gruppen inte hade en gemensam uppfattning hur estimering av *tasks* gick till. Enkäten besvarades av varje deltagare individuellt och resultatet från den finns i bilaga A.

Enkäten visar på att projektet inte kommer upp i nivå två av CMM. Enligt Key Practices of the Capability Maturity Model[4] Aktivitet 9 måste projektet estimera tidsåtgång enligt en dokumenterad process. XP-processen har ett sådant moment i varje iteration dock påvisar enkätsvaren att detta inte görs i detta projekt. Enligt CMM skall även tidsåtgången omestimeras vid eventuella ändringar av kraven vilket inte heller görs i detta projekt.

Aktivitet 5 i CMM säger att de tidsuppskattningar som görs skall dokumenteras och eftersom, enligt tidigare resonemang, det inte finns några estimeringar kan inte heller dessa dokumenteras.

Detta innebär att projektet Orfeus inte kan uppnå nivå två (2) av CMM och befinner sig därför automatiskt i nivå ett (1).

Produktivitet

För att bestämma produktiviteten för projektet Orfeus vill man kunna mäta antalet skrivna programkodsrader per mantimmar och iteration. Man vill då också kunna jämföra skillnaden mellan två iterationer. Enkätsvaren visar dock, enligt bilaga A, på att det finns en kraftig oenighet om vad som har varit en iteration i projektet. Därför går det inte heller tydligt att säga vad som har varit en iteration. Genom att granska progressrapporterna från projektgruppen kommer denna rapport istället att beräkna programkodsrader per mantimmar och progressrapportstillfälle.

Hittills i projektet har det skrivits två (2) progressrapporter. Den ena rapporten är för vecka 45 till och med vecka 47 och den andra för vecka 48 till och med vecka 50. Detta kan användas som två mätpunkter eftersom projektet använder ett versionshanteringsystem för programkoden vilket gör det möjligt att få fram information om programkodens utseende vid dessa tillfällen.

Vecka 45 till 47

Under denna tid är den sammanlagda arbetstiden 351 timmar. Från den initiala versionen av programmet fram till den version som var tillgänglig i slutet av vecka 47 (1.2) skiljer det 266 programkodsrader. Det har alltså tillkommit 266 programkodsrader. För att bestämma hur mycket programkod de gjort per mantimme beräknas

$$\frac{programkodsrader}{mantimmar} = \frac{266}{351} = 0{,}758$$

Vecka 48 till 50

Under denna tid är den sammanlagda arbetstiden 399,5 timmar. Från versionen som fanns tillgänglig i vecka 48 (1.2) till den som fanns vecka 50 (1.27) skiljer det sig 216 rader. Det har alltså tillkommit 216 programkodsrader. Än en gång bestäms hur mycket programkod de gjort per mantimme genom

$$\frac{programkodsrader}{mantimmar} = \frac{216}{399{,}5} = 0{,}541$$

Resultat

Med detta sätt att mäta har produktiviteten minskat mellan dessa två mätintervall. Det är synligt att vecka 48 till 50 haft en lägre produktivitet än vecka 45 till 47. Dock är inte detta hela sanningen eftersom ingen hänsyn har tagits till ifall gammal kod har fått kasseras eller svårhetsgraden på de implementerade funktionerna. En annan tänkbar anledning till minskningen av produktiviteten kan vara deltagarnas motivation, denna kan tänkas ha varit högre i inledningen av projektet. Det vill säga de första tre veckorna.

Detta kan dock ge en fingervisning om deras produktivitet.

Återanvändningsgrad

Projektet Orfeus är en vidareutveckling av en tidigare mjukvara. Programkoden för denna mjukvara har tillhandahållits projektet vid start. Detta skulle enkelt kunna klassas

som återanvändning av programkod. I programkoden till projektet används även färdiga bibliotek för exempelvis diskhantering och skärmutskrift.

I denna rapport klassas återanvändning enbart som den programkod projektet tillhandahölls vid start. Att i denna definition även inkludera de bibliotek som används skulle resultera i att den producerade programkodsmängden skulle bli liten i förhållande till den tillhandahållna. Enligt Poulin[3] skulle detta vara en rimlig bedömning. Måttet på återanvändningen kommer att vara en kvot mellan antalet återanvända programkodsrader och det nuvarande antalet programkodsrader.

Att göra denna definition innebär att all återanvänding representeras av tillhandahållen programkod. Det krävs då att inte allt för mycket av denna programkod har förändrats med tiden.

Projketet Orfeus har ett versionshanteringsystem där man finner samtliga versioner av programkoden som utvecklats. Här återfinns den ursprungliga versionen av programkoden som tillhandahölls projektet vid start, men även den senaste versionen av programkoden. Med hjälp av mjukvaran SourceMonitor[1] bestämdes antalet programkodsrader i både den ursprungliga versionen samt den senaste versionen (1.27). SourceMonitor passade analysens behov bra eftersom programmet mäter en programkodsrad på samma sätt som tidigare definierat i denna rapport.

Mätningen gjordes den 14 december 2006 på version 1.27 av programkoden och var då på 995 programkodsrader. Den ursprungliga programkoden, version 1.1, är på 513 programkodsrader. Därför blir aktuella återanvändningen 51,6% av den totala kodmängen.

$$\frac{\#NCLOCv.1.27}{\#NCLOCv1.1} = \frac{513}{995} = 0{,}516$$

En återanvändningsgrad på cirka 50% verkar inte helt orimligt med tanke på att projetet bygger vidare på en befintlig produkt. Dessutom har inte projektet pågått så lång tid och en del av denna tid har gått till att sätta sig in i befintlig programkod. Detta borde innebära att de under denna tid inte har hunnit skriva någon större mängd programkod.

Designstruktur

Projektet Orfeus programkod är samlat i en enda programkodsfil. För att mäta cohesion och coupling behövs definierade moduler för programmet. Eftersom alla funktioner för programmet är samlat i en enda programkodsfil kan detta tolkas som att det finns endast en modul i programmet. Denna tolkning av moduler ger projektet Orfeus i stort sett det värsta tänkbara resultatet för design.

Coupling

Nivån av coupling utgörs av 6 steg. Dessa steg klassificeras från bäst couplingsgrad till värst couplingsgrad. Om en modul inte har någon coupling räknas till den allra bästa couplingsgraden. Dessa steg finns definierade i boken Software Metrics[2].

Programkoden tolkas nu som en modul enligt ovanstående anledning. Resultatet från en sådan tolkning blir mycket intuitiv. Eftersom det enbart finns en modul i

programmet kan denna inte kommunicera med någon annan modul vilket resulterar i att det inte finns någon coupling.

Cohesion

Nivån av cohesion utgörs av 7 steg. Dessa steg klassificeras från bästa till värsta. Om en modul inte har någon cohesion räknas till den värsta av dessa steg. Dessa steg finns definierade i boken Software Metrics[2].

Programkoden tolkas nu som en enda modul. Resultatet från denna tolkning är även den ganska intuitiv. Eftersom programmet utför ett antal olika uppgifter måste dess cohesion vara dålig. För att kunna ta reda på vilket steg av cohesion programmet befinner sig på måste hela modulens kod utvärderas. Modulen består av ett antal C-funktioner, samtliga C-funktioner har inte utretts. Istället har C-funktioner som misstänkts ge det högsta steget av cohesion granskats närmre. Bland dessa C-funktioner finner vi `ceildiv()` och `strtoul_ck()` som har helt orelaterad funktionalitet. Med orelaterade funktioner menas två eller flera funktioner vars funktionalitet inte bidrar till de andra funktionerna. Det finns dessutom med säkerhet fler C-funktioner som har dessa egenskaper, men då redan det värsta steget av cohesion infunnit sig finns ingen anledning att analysera dessa.

Detta resulterar i att hela modulen uppnår cohesionsteget *coincedental*.

Resultat

Om tolkningen att ingen coupling är låg coupling görs så får vi så kan man mycket väl se sambandet mellan coupling och coheison. Nivån av coupling är så låg den kan bli vilket resluterar i att vi har mycket dålig cohesion. För coupling fann vi ingen nivå och för coheison fann vi nivån *coincedental*.

Feldensitet

Projektet Orfeus saknar bestämda iterationer, enligt tidigare resonemang, och därför är det inte möjligt att mäta feldensiteten per iteration. Istället kommer feldensiteten att mätas på de mätpunkter som återfinns i projekgruppens progressrapporter. Det är inte heller möjligt att mäta om feldensiteten skiljer sig mellan olika *tasks*. Detta eftersom det inte finns några tydliga *tasks* och de som kan tolkas som *tasks* varken är avslutade eller väldefinierade.

Feldensiteten beräknas istället på totalt antal rader programkod som fanns vid iterationens slut och det antal fel som ännu inte var åtgärdade vid tidpunkten. I detta fall räknas även den återanvända programkoden med, detta eftersom det kan påträffas fel även i denna del av koden och som inte var kända tidigare.

Vecka 45 till 47

När denna period är avslutad fanns det, enligt buggrapporterna, två (2) icke åtgärdade fel. Vid periodens slut bestod programmet av 779 programkodsrader (version 1.2). Detta ger oss feldensiteten för denna version som

$$\frac{\#\,fel}{programkodsrader} = \frac{2}{779} = 0{,}0026$$

Vecka 48 till 50

När denna period är avslutad fanns det, enligt buggrapporterna, fem (5) icke åtgärdade fel. Vid periodens slut bestod programmet av 995 programkodsrader (version 1.27). Detta ger oss feldensiteten för denna version som

$$\frac{\#\,fel}{programkodsrader} = \frac{5}{995} = 0{,}005$$

Resultat

Det som går att härleda från denna information är att feldensiteten har ökat mellan de två mätintervallen. Vad detta beror på är svårt att säga. Det är tänkbart är att detta beror på att den inledande programkoden var enklare att skriva, vilket även den högre produktiviteten skulle kunna vara ett tecken på, men även att felsökningen inte var lika omfattande i den inledande fasen.

Avslutning av projektet

Eftersom detta projekt drivs enligt XP-metoden är det i sin natur näst intill omöjligt att förutbestämma den slutgiltiga produktens omfattning. Enligt XP-modellen är produkten färdig när projektets budget är slut. Detta fungerar eftersom det i slutet av varje iteration finns en fungerade version av produkten.

Som tidigare nämnt finns det inga tydliga iterationer definierade för Orfeus. Eftersom det inte finns några väl definierade slut för iterationerna så finns inte heller någon bestämd punkt då programmet skall befinna sig i ett körbart tillstånd. Projektet har också ett antal *storyer* varav ingen av dem ännu är avklarade (den 14 december 2006).

Avslutning inom avsatt tid

Som tidigare sagt hade projektet, om de följt XP-processen, med största sannolikhet varit klart vid projekttidens slut. De mätdata som finns tillgängliga säger ganska klart och tydligt att de inte hinner i tid. Som tidigare nämnt finns inga riktiga iterationer i projektet och fokus för rapporten ligger istället på de två progressrapporter som inlämnats.

De två progressrapporter som finns inlämnande beskriver veckorna 45 till och med 47 samt veckorna 48 till och med 50. Under hela denna tid (vecka 45 till och med 50) har det tillkommit nio (9) kort i systemet. Enligt källan KAUNET[5] på SourgeForge är endast fyra (4) av dessa tilldelade till projektmedlemmar. De andra korten är inte tilldelade någon. Av dessa fyra (4) kort har inget ännu avslutats.

Genom att ta hänsyn till dessa faktorer verkar det inte speciellt troligt att projektet hinner klart i tid. Hittills har inget av de kort som tilldelats någon slutförts och dessutom finns det flertalet kort som inte påbörjats. Det återstår en progressrapport innan deadline och det är tänkbart att det tillkommer nya *storyer*.

Denna mätning och uppskattning har relativt hög osäkerhet. Inga *storyer* eller *tasks* har tidsuppskattats. Det kan också vara så att ett antal av de *storyer* som finns på KAUNET[5] har avslutats utan att information om detta har lagts upp på webbplatsen. Det finns dessutom en risk för att en del av de *storyer* som finns har tilldelats någon utan att detta har uppmärksammats på sidan.

Att situationen är sådan kan inte vara bra för projektet. Det ger inte en ögonblicksbild över arbetsytan för att se vad som pågår och vad som kommer att ske härnäst, vilket XP föreskriver. XP-metoden säger dessutom att den slutgiltiga produktens omfattning inte är känd eftersom omfattningen styrs av hur mycket projektmedlemmarna hinner med på den budgeterade tiden.

Uppskattad slutlig programkodmängd

Som underlag till uppskattningen för den slutgiltiga programkodsmängden används den nuvarande programkodsmängden räknat i programkodsrader. Från denna har återanvändningen tagits bort på grund av att projektet bygger vidare på befintlig kod. Det är inte troligt att de återanvänder en lika stor mängd kod igen. Detta ger att den mängd kod som kommer att skrivas baseras på den aktuella produktiviteten samt den tid som projektet har kvar.

Det antas att projektet fortskrider med den nuvarande produktiviteten, som tidigare visats att vara 0,541 programkodsrader per mantimme. Hittills i projektet har 750,5 mantimmar förbrukats vilket ger 449,5 mantimmar kvar. Med i beräkningar finns även den nuvarande programkodsmängden vilken är 995 programkodsrader.

Detta skulle ge en uppskattad programkodsmängd på
$995 + 0{,}541 \cdot 449{,}5 = 995 + 243{,}1795 = 1238$ programkodsrader.

Den uppskattade slutgiltiga programkodsmängen på 1200 programkodsrader är inte speciellt tillförlitligt då delvis uppskattningen av produktiviteten och den fortsatta produktiviteten i sig är osäker. Dessutom finns det ett antal andra aspekter som kan påverka antalet programkodsrader i den slutgiltiga versionen. Exempelvis skulle sista veckan enbart kunna användas till att rätta fel i mjukvaran eller så måste flera funktioner kasseras och programmets storlek sjunker. Dessutom finns ingen information om de *storyer* som nu är öppna har mycket kvar innan de avslutas och om de nya *storyerna* som skrivs är av samma omfattning som de tidigare.

Problem

Ingen estimering

Ett problem som påträffades vid granskning av de *storyer* som fanns för projektet var att ingen av dem var estimerade med tid. Det fanns alltså inget sätt att veta hur länge de hade planerat att arbeta med en *story*. Det fanns då inte heller något sätt att se när *storyn* beräknades färdig.

Iterationer

Det misstänktes redan vid första besöket hos projektgruppen att det här med XP som en iterativ process var det ingen som riktigt tog på allvar. Istället var de flesta oense om vad en iteration i deras projekt var och när de började och slutade. För att kunna påvisa detta tillverkades en anonym enkät som deltagarna i projektet fick svara på. Resultetet från enkäten finns i bilaga A och visar på att projektgruppen inte kan arbeta enligt iterationer då gruppen inte är ense om hur många iterationer de gjort och när de börjat

och slutat. Att det inte fanns några iterationer har gjort några frågeställningar svårbesvarade.

Inga avslutade storyer

Inga av de kort som fanns på KAUNET[5] var avslutade den 14 december 2006 då mätdata togs. Detta gjorde att det blev problem med att bestämma hur mycket de har hunnit med hittills.

Coheison och coupling

Eftersom programkoden enbart består av en fil har det varit lite klurigt att komma på något vettigt för cohesion och coupling. Att tolka det som en enda modul ger en mycket enkel tolkning och ett klart resultat. Informationen från denna känns inte direkt värdefull, inte för coupling i alla fall.

Det fanns en idé att gruppera samtliga funktioner. Dock har detta varit svårt för både oss och projektmedlemmarna. Tolkningen skulle också enbart resultera i att vi får en ”tänkt” design. Designen ser inte ut så här i dagsläget och vi mäter då på något som inte finns.

Referenser

1) SourceMonitor – CampWood Software (www.campwoodsw.com)
2) Software Metrics, Second Edition Revised Printing – Norman E. Fenton, Shari Lawrence Pfleeger, PWS Publishing Company
3) Mesuring Software - Reuse, Jeffrey S. Poulin, Addison-Wessley
4) http://www.sei.cmu.edu/pub/documents/93.reports/pdf/tr25.93.pdf
5) KAUNET – http://sourgeforge.net/projects/kaunet

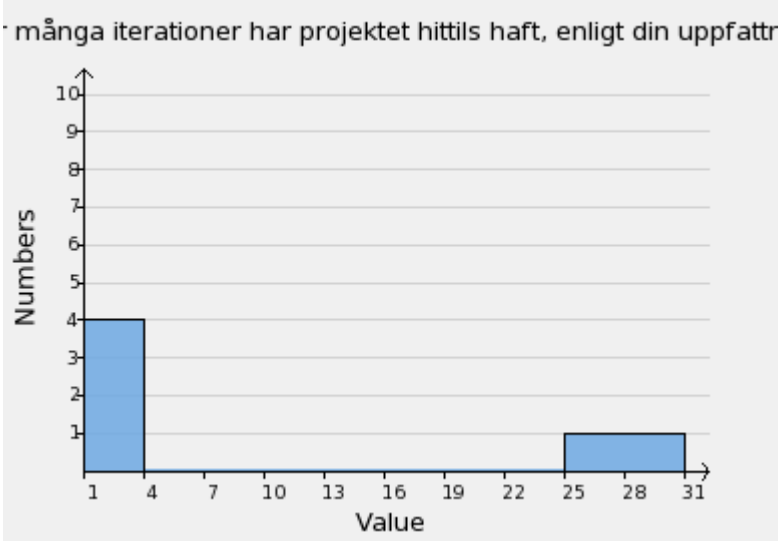
Bilaga A - Enkät

Substantial Facts

Number of questions:	7
Number of answers:	6
Fully answered:	6
Number of assigned users:	6
Answer frequency:	100.0%

Question	Avg. score	Median	Std. deviation
5. Vi följer Uppdragsspecifikationen som finns på webbplatsen.	66.7	75.0	40.8
6. Vi estimerar tidsåtgång för tasks i projektet.	-8.3	-25.0	73.6
7. Vid ändring av en befintlig task (uppdaterade krav etc) estimerar vi om tidsåtgången.	-33.3	-50.0	75.3

1. Hur många iterationer har projektet hittills haft, enligt din uppfattning?



2. Hur många uppgifter (tasks) har ditt par avslutat i senaste iterationen?

3. Hur många timmar arbetar du, i genomsnitt, per vecka?