# Coaching av programvaruteam (EDA270)
# Djupstudie: Användbarheten av commit comments

Robert Andersson Haraldsson
D05, Lunds Tekniska Högskola
dt05ra1@student.lth.se

Marie Li Korse
D05, Lunds Tekniska Högskola
dt05ml9@student.lth.se

18 februari 2008

# Innehåll

**1 Inledning** **3**
1.1 Bakgrund . . . 3
1.2 Frågeställning . . . 3
1.3 Hypoteser . . . 4
1.4 Undersökningsmetod . . . 4
1.5 Resultat . . . 4

**2 Spårbarhet** **4**
2.1 Vikten av commit comments . . . 5

**3 Inbjudande commit comments** **5**

**4 Commit comment first** **6**

**5 Resultat** **6**
5.1 Jämförelse av resultat mellan användning av mallar och ingen anvädning av mallar . . . 7
5.2 Jämförelse mellan commit comments i början av projektet och i slutet av projektet . . . 8
5.3 Enkätsvar . . . 8

**6 Slutsatser** **9**

**7 Referenser** **12**

**Sammanfattning**

Rapporten behandlar vikten av commit comments i projekt, en del allmänt om spårbarhet samt en studie av hur commit comments kan förbättras. Studien görs på tre olika XP-team, från kursen *EDA260 - Programvaruutveckling i grupp*, som får prova ut ett antal mallar som vi har utvecklat i syfte att just standardisera och förbättra commit comments. Vidare testas en idé som kallas *commit comment first*, som innebär att man skriver sin commit kommentar innan man börjar implementera.

# 1 Inledning

Detta är en djustudie som behandlar vikten av commit comments, samt en undersökning om hur dessa kan förbättras, gjord vid Lunds Tekniska Högskola under kursen *EDA 270 - Coaching av programvaruteam.*

Djupstudien behandlar, efter en kort beskrivning av bakgrunden, frågeställning, hypoteser, undersökningsmetodiken samt en sammanfattning av resultatet, först spårbarhet i allmänhet. Efter detta stycke följer en mer detaljerad beskrivning om hur commit comments relaterar till spårbarhet. Därpå följer ett stycke som berör hur commit comments kan göras bättre och i allmänhet mer omtyckta av utvecklare, samt en liten presentation av material till undersökningen såsom mallar. Commit comment first, som är det sista avsnittet innan resultatet och slutsatsen av undersökningen presenteras, behandlar ett alternativt sätt på hur en commit comment skulle kunna bidra till ett projekt mer än enbart i form av dokumentation.

## 1.1 Bakgrund

När olika personer arbetar med ett projekt som involverar programmering, behövs ett gemensamt utrymme för koden. Varje enskild person som arbetar med koden arbetar dock med en kopia av koden för att förhindra eventuella misstag och krockar. När denna person beslutar sig för att skicka in sina ändringar till det gemensamma utrymmet, kallas det att han eller hon gör en commit. För att underlätta för övriga utvecklare i projektet, skickas en kort kommentar med diverse nödvändig information med i samband med personens commit. Denna kommentar är det som benämns som commit comment.

Vår studie av commit comments har gjorts på ett XP-projekt[1] med studenter från kursen *EDA260 - Programvaruutveckling i grupp*, med versionshanteringsverktyget CVS. Praktiska skillnader som detta har medfört för vår studie är att man programmerar i par i XP-projekt tillskillnad från de flesta andra typer av modeller som används i projekt. Vad det gäller skillnader mellan olika versionshanteringsverktyg, finns det olika mycket stöd och behov av commit comments. I CVS registeras exempelvis automatiskt den personen vars konto ändringen skickas ifrån som författare till kommentaren. Andra verktyg har ännu mer stöd för ändringar som skickas in, som i SVN där även vilka filer som ändrats automatiskt sparas. I dessa fall är finns det alltså inte behov av samma information i kommentaren.

## 1.2 Frågeställning

Med vår djupstudie vill vi undersöka följande frågor:

- På vilket sätt gynnar commit comments ett programvaruprojekt och vilka parametrar har störst betydelse för att commit comments ska vara användbara?
- Hur påverkar projektets storlek vikten av genomtänkta commit comments?
- På vilket sätt kan man underlätta skrivandet av värdefulla commit comments? Har de mallar vi utformat bidragit till bättre commit comments?
- Har det underlättat arbetet att skriva commit commets före själva implementeringen?

[1] XP är en förkortning av projektmodellen *eXtreme Programming*

### 1.3 Hypoteser

1. Kvaliteten på commit comments är lägre om man inte använder något standariserat sätt att skriva dem på, det vill säga utan mallar.
2. Vid mindre projekt, kommer informationsrika commit comments inte att behövas i samma utstäckning som i större projekt.
3. Mallar ger en bättre stuktur på commit comments och underlättar för utvecklarna.
4. Antalet värdelösa commit kommentarer är fler under andra halvan av XP-projektet än första, då utvecklarna har lärt känna varandra bättre och inte orkar ta kursen på lika stort allvar.
5. *Commit comments first* är krävande, men underlättar vid implementeringen och hjälper utvecklarna att formulera ett konkret mål.
6. *Commit comment first* leder till god designen och en bra taskindelning.

### 1.4 Undersökningsmetod

I syfte att underlätta skrivandet av användbara commit comments, har vi utformat tre olika mallar åt utvecklarna. För att undersöka på vilket sätt mallarna kan utnyttjas maximalt och ifall de överhuvudtaget har bidragit till bättre commit comments, har de använts på tre olika sätt vid tre olika tillfällen, eller rättare sagt under tre olika iterationer. Under den första iterationen har vi inte gett utvecklarna någon form av information om hur deras commit comments bör utformas. Inför den efterföljande iterationen gav vi dem tre olika mallar för olika typer av utveckling som de blev ombedda att använda sig av. Den sista iterationen var uppgiften för dem att skriva sin commit comment före själva implementeringen, vilket vanligtvis sker sist. Mallarna är utformade efter olika slags utveckling, nämligen implementering, buggfixande och refaktorisering, se Appendix A. Efter undersökningen ombeddes utvecklarna även att fylla i en undersökningsenkät. Även denna finns i Appendix A.

### 1.5 Resultat

Att på ett standariserat sätt skriva mallar, har enligt vår studie visat sig vara användbart. Det gör att utvecklarna får med all viktig information vid en commit comment, så att nästa utvecklare får en bra överblick av vad som har ändrats. Vidare går det både snabbare och enklare och blir bättre kvalitet på commit-kommentarerna när mallarna tillämpas. Mallar är alltså ett bra verktyg för att förbättra kvaliteten och göra commit comments användbara för tillbakablickar, spårning och sökning i koden, medan *commit comment first* är en metodik som inte bidrog med speciellt mycket och var svårtillämpad.

## 2 Spårbarhet

Spårbarhet kan beskrivas som ett fenomen som hanterar dokument och dess relationer till andra dokument och entiter. Spårbarhet är ett sätt att hantera spårning av dokument och andra artifakter som skapas och ändras under tiden ett projekt fortlöper. [2]

För att kunna ge goda prediktioner om framtiden och för att verifiera kravspecifikationen för ett projekt, krävs ett gott underlag på hur det gått fram till den tidpunkten. Bland detta underlag ingår all form av dokumentation vilket gör spårbarhet möjligt. Med hjälp av detta kan krav kopplas ihop till kod samt information om olika tidsaspekter och vilka personer som har varit involverade. Detta gör att man lättare kan verifera att krav faktiskt är implementerade och att man lätt i efterhand kan kolla upp detaljer kring ett krav. Spårbarhet behövs även för att man lättare ska kunna estimera kostnader, i form av nerlagd tid och resurser, för kommande ändringar inom ett projekt. Det underlättar även för att approximera arbetet vid en specifik ändring. [2][3]

Spårbarheten är ett medel för bättre informationshantering och kan underlätta kommunikation som sträcker sig över en längre tid, detta genom att personer blir påminda om vad som tidigare har blivit gjort av god dokumentation. Detta gör det möjligt för en person att kolla upp information från tidigare

dokument, vilket i sig är en slags kommunikation.

Spårbarhet är ingen lätt process att följa. Det finns inga färdiga riktlinjer eller något specifikt sätt gällande hur det ska genomföras. Kostnaden, mantimmarna som krävs för att skapa och underhålla alla dokument, för spårbarheten måste dessutom vägas emot vad det faktiskt ger. I mindre projekt är kanske inte all information lika väsentlig att dokumentera eftersom projektet inte är lika långvarigt. [2][3]

## 2.1 Vikten av commit comments

Commit comments är klassat som förmedlande kommunikation (mediated communication). Det vill säga information som är tillgänglig för en individ via olika projekttillgångar. Förmedlande kommunaktion brukar vara passiv envägskommunikation som har i syfte att öka en individs medvetenhet och kännedom om vad som händer och har skett. Detta för att det kan hjälpa en person att planera sina aktiviteter med övrig personal. Det vill säga commit comments ger kort information om vad en utvecklare har gjort och tänkt, vilket underlättar för nästföljande utvecklare. Vidare om det är några frågetecken, kan personen utnyttja kommentaren till att ta ytterligare kontakt med den som gjort committen. [4]

Commit comments är en lätt och smidig typ av dokumentation av ändringar gjorda i projekt där versionshanteringsverktyg används. Commit comments har fördelen av att ofta ge en kort summering av vad som har ändrats och hur utvecklaren i fråga har tänkt. Commit comments ger på så sätt information som är skild från exempelvis dokumentering direkt i koden. Commit comments har en del fördelar gentemot annan koddokumentation. Den är dels skriven på ett naturligt språk och länkas direkt till en versionsändring av en fil. Detta gör att den är lätt att komma åt och ofta ger väsentlig information, såsom vad som nyligen ändrats. Med bra commit comments så blir det alltså lättare att hitta rätt version av filen man söker och man behöver inte bläddra igenom lika mycket kod. [1]

En annan fördel är att utvecklarna med stor sannolikhet hellre skriver en commit comment än dokumentation i koden. Detta kan bero på att man ser commit-kommentaren som en chans att beskriva vad man ändrat och hur man tänkt, och att man inte har lika stora krav på kvaliteten av kommentaren. Kvaliteten behöver inte vara lika stor, eftersom det inte är lika många som behöver se commit-kommentaren. Detta innebär pressen minskar och det blir naturligt att skriva commit comments. Vidare fångar commit comments dokumentation som eventuellt inte beskrivits någon annanstans, sådant som buggfixning och refaktorisering. Om man fixat en bugg skriver man gärna inte det som en kodkommentar, men commit comments ger en chansen till att berätta att buggen är fixad och hur man gjorde det. [1]

Att kontrollera vem som har ansvaret för ett visst område ingår i spårbarhet. Detta underlättas av commit comments, då varje person som implementerar något, även skickar med en kommentar om vad som gjorts och av vem. Inom XP är dock alla ansvariga för all kod. I praktiken innebär detta inte någon större skillnad gällande vilken information kommentarerna bör innehålla men utvecklarna som författat dem är inte ensamt ansvariga för den kod de implementerat. Dock kan man ändå ha stor nytta av att se vem som har gjort ändringarna, i synnerhet om en annan utvecklare vill fortsätta arbeta med koden. [2]

# 3 Inbjudande commit comments

Ibland finns det en viss motvilja bland utvecklare att dokumentera vad som har gjorts, då det anses minska det ”viktiga” arbetet, nämligen implementeringen. Detta kan bland annat leda till mindre utförliga eller oseriösa commit comments och ibland helt enkelt inga alls (Se Resultat Utan mallar för exempel). Detta kan leda till problem senare, i synnerhet ifall det är ett större projekt som fortlöper under en lång tidsperiod. Ifall en person som är väldigt insatt i en viss del av programmet plötsligt slutar och inte har fört någon form av dokumentation, kan det bli väldigt problematiskt för efterträdaren.

Den främsta anledningen till motviljan är ofta känslan av att det inte ger någonting, och att det är jobbigt. För att motivera utvecklarna till att ändå skriva commit comments, är exempelvis att påtala deras betydelse samt försöka minska arbetsbördan som krävs för att skriva dem. Det förstnämnda är kanske inte alltför effektivt ifall utvecklarna inte själva tror eller känner att de ger någonting, men att

minska och konkretisera innehållet av en commit comment och dämed minska arbetsbördan kan göras med exemeplvis mallar.

Mallarna är utformade så att all viktig information ska komma med, men samtidigt vara korta och lätta att skriva. För att optimera detta har vi utformat individuella mallar för olika typer av utveckling, då olika typer av information är olika viktig beroende på vad som har utförts. Nedan följer en förklaring till varför vi anser att respektive rubrik i mallarna tillför något till projektet (Orginal mallarna finns i Appendix A).

- *Utvecklare* - Att veta vem som är ansvaring för det som har gjorts är viktigt dels för att någon ska kunna stå till svars för det som utförts, och dels när någon annan behöver ytterligare information om ändringarna.
- *Ändringar har gjorts i följande klasser/metoder, Implementationer och Ändringar gjorda i klasser, Ändringar har gjorts i följande klasser* - Vilka ändringar som gjorts och var detta har skett, är den information som efterföljande utvecklare kan använda sig av för att kunna skaffa sig en överblick av vad som har hänt och var samt som hjälpmedel för att snabbt orientera sig i koden vid oklarheter.
- *Story/Task implementerad, Kort beskrivning av buggen* - Ifall en utvecklare vill söka efter hur en viss bugg har blivit fixad eller hur man har löst en viss story, underlättar det enormt om dessa rubriker finns med i kommentaren. Denna information är väldigt svår att hitta både i koden och i exempelvis javadokumentation.
- *Kort beskrivning av fixen* -Att föklara på vilket sätt man har löst en bugg, kan vara väldigt användbart för andra utvecklare ifall de skulle stöta på ett liknande problem. Istället för att gå in och titta på de klasser författaren har ändrat i, finns en kort summering under den här rubriken.
- *Typ av refaktorisering* - Att veta vilken typ av refaktorisering som gjorts, ökar förståelsen för hur saker har förändrats och blir mindre tidskrävande att sätta sig in i. .

# 4 Commit comment first

*Commit comment first*, innebär att man redan innan man börjar implementera en story/task ska ha tänkt ut och skrivit ner en commit comment. Denna commit comment ska man sedan titta på när man har implementerat klart. Om man har implementerat uppgiften som man först tänkte sig ska det bara vara att kopiera in kommentaren som commit comment, annars får man ändra på den när man sedan commitar. Det är även tänkt att man ska använda sig av färdiga mallar vid *commit comment first*. Även till detta användes mallarna i appendix A.

Tanken bakom *commit comment first* är att utvecklarna inte ska börja implementera direkt, utan att de först tänker igenom hur de vill lösa problemet och var ändringarna behövs göras. Tanken är att utvecklarna lättare ska inse om det behövs nya klasser, om designen behövs ändras och vilka metoder som ska ändras eller skapas. Detta är tänkt att göra det lättare för utvecklarna att hitta en röd tråd och ha något att ta hjälp av och då inte på samma sätt köra fast vid implementeringen. *Commit comment first* låter även utvecklaren få en klar överblick av hur en story kan lösas och på så sätt bidra till en bättre taskindelning för den.

Att tänka ut hur man ska lösa en uppgift på förhand kan vara väldigt svårt och ibland är det lättare att bara angripa problemet direkt, vilket i många fall är också är fullt tillräckligt. Vi vill dock prova hur det fungerar om utvecklaren alltid tvingas fundera kring problemet före implementationen och ifall det faktiskt ger någonting. Det vill säga om man gör en stor uppoffring i början kanske man får tillbaka detta i form av en god design och ett bra flyt vid implementationen.

# 5 Resultat

Studien är gjord på studenter från kursen *EDA260 - Programvaruutveckling i grupp* år 2007 och år 2008. Vid jämförelsen mellan användning av mallar och ingen anvädning av mallar, baserades resultaten

på tre projektgruppers första och andra iteration. Första iterationen gavs utvecklarna inga former av restriktioner eller anvisningar, medan de fick i uppgift att följa mallarna i Appendix A under iteration två.

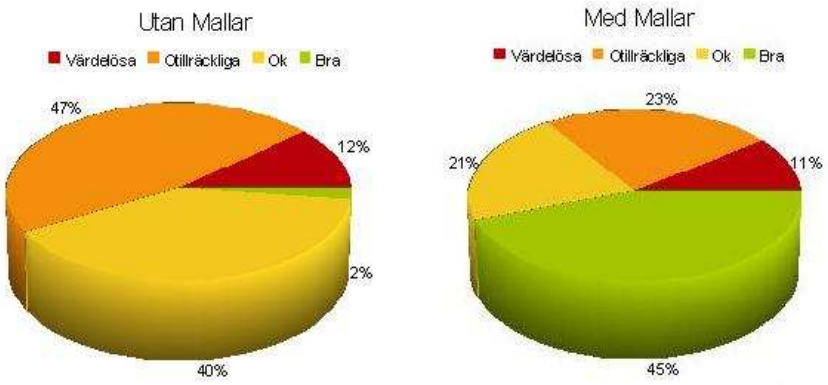
Vid jämförelse mellan commit comments i början av projektet och i slutet av projektet, baserades resultaten på två olika projektgruppers samtliga sex iterationer. Dessa grupper fick inga anvisningar eller riktlinjer unde iterationerna.

Kommentarerna har vi värderat till någon av följande fyra kategorier: Värdelösa, Otillräckliga, Ok eller Bra. Kriterierna för att hamna i de olika grupperna är persenterade i listan nedan. För att hamna i kategorin värdelös, är antingen kommentaren helt tom eller utan relevant information. Exempel på sådana kommentarer är 'Middag för 2." och "From russia with multibyte love" tagna från projekt 03 år 2007 i klassen enduro.gui/RegisterGUI.java.

- *Otillräckliga* - Vilka ändringar som har gjorts eller var ändringarna har skett ska finnas med.
- *Ok* - Vilka ändringar som har gjorts och var ändringarna har skett ska finnas med.
- *Bra* - Vilka ändringar som har gjorts och var ändringarna har skett samt vilka utvecklare som har arbetat med uppgiften ska finnas med.

## 5.1 Jämförelse av resultat mellan användning av mallar och ingen anvädning av mallar

Statistiken nedan är baserade på de commit comments som fanns i de tre klasser med flest olika versioner från de tre gruppernas repositorier, alltså totalt nio olika klasser.




| Metod | Värdelösa | Otillräckliga | Ok | Bra |
|---|---|---|---|---|
| Utan mallar | 5 | 20 | 17 | 1 |
| Med mallar | 5 | 11 | 10 | 21 |

## 5.2 Jämförelse mellan commit comments i början av projektet och i slutet av projektet

Statistiken nedan är baserade på de commit comments som fanns i de tre klasser med flest olika versioner från två gruppers repositorier från 2007, alltså totalt sex olika klasser.



| Period | Värdelösa | Otillräckliga | Ok | Bra |
|---|---|---|---|---|
| Iteration 1-3 | 11 | 47 | 34 | 1 |
| Iteration 4-6 | 25 | 42 | 19 | 0 |

## 5.3 Enkätsvar

Sammanställning av svaren på frågorna i Appendix A. Bilden finns även där för att man enklare ska se frågorna samtidigt som man observerar resultatet.

Representativa motiveringar till frågor i enkäten:

| Fråga | Ja[st] | Nej[st] | Vet ej[st] |
|---|---|---|---|
| 1 | 4 | 14 | 0 |
| 2 | 12 | 6 | 0 |
| 3 | 15 | 3 | 0 |
| 4 | 18 | 0 | 0 |
| 5 | 14 | 3 | 1 |
| 6 | 3 | 15 | 0 |
| 7 | 15 | 3 | 0 |
| 8 | 7 | 10 | 1 |
| 9 | 14 | 4 | 0 |
| 10 | 2 | 16 | 0 |
| 11 | 14 | 1 | 3 |
| 12 | 3 | 13 | 2 |
| 13 | 7 | 6 | 5 |
| 14 | 4 | 9 | 5 |
| 15 | 0 | 14 | 4 |
| 17 | 7 | 3 | 10 |

| Fråga | Ja[%] | Nej[%] | Vet ej[%] |
|---|---|---|---|
| 1 | 0,22 | 0,78 | 0 |
| 2 | 0,67 | 0,33 | 0 |
| 3 | 0,83 | 0,17 | 0 |
| 4 | 1 | 0 | 0 |
| 5 | 0,78 | 0,17 | 0,06 |
| 6 | 0,17 | 0,83 | 0 |
| 7 | 0,83 | 0,17 | 0 |
| 8 | 0,39 | 0,56 | 0,06 |
| 9 | 0,78 | 0,22 | 0 |
| 10 | 0,11 | 0,89 | 0 |
| 11 | 0,78 | 0,06 | 0,17 |
| 12 | 0,17 | 0,72 | 0,11 |
| 13 | 0,39 | 0,33 | 0,28 |
| 14 | 0,22 | 0,5 | 0,28 |
| 15 | 0 | 0,78 | 0,22 |
| 17 | 0,39 | 0,17 | 0,56 |

| Omdöme | Utan mallar[st] | Mallar först[st] | Mallar sist[st] |
|---|---|---|---|
| Bäst | 1 | 3 | 14 |
| Nästbäst | 11 | 3 | 4 |
| Sämst | 6 | 12 | 0 |

| Omdöme | Utan mallar[%] | Mallar först[%] | Mallar sist[%] |
|---|---|---|---|
| Bäst | 0,06 | 0,17 | 0,78 |
| Nästbäst | 0,61 | 0,17 | 0,22 |
| Sämst | 0,33 | 0,67 | 0 |

Figur 1: Enkätsvar

- *Fråga 3* - ”Nja, svårt att säga för man kommer ihåg allt man gör i ett så här litet projekt”, ”Nja, de talar ju om vad som ändrats och vem som gjort det. Men om man inte förstår sig på ändringen 20 år senare lär inte den som gjorde det göra det heller.”
- *Fråga 5* - ”Ja. Man vet vad man ska ta med i sina comments och det går fortare att skriva dem Det har gjort det lättare att komma på hur jag ska formulera mig när jag skriver en commit comment. Dessutom har det hjälpt till med att göra kommentarerna tydligare och mer lättöverskådliga.”
- *Fråga 6* - ”Om man skulle göra en snabb commit av en mockup så kändes det onödigt att skriva att en story var implementerad.”, ”Nej, det har till och med gjort det lättare att skriva bra commit comments.”
- *Fråga 8* - ” Jag har inte behövt anstränga mig mer eftersom det är lättare att fylla i en mall än att behöva komma på kloka kommentare själv.”
- *Fråga 9* - ”Ja. Nu får man med det som är viktigt, vilket man kanske glömde innan.”
- *Fråga 10* -” Nej inte mer stöd, bara ordentliga regler för hur dom ska skrivas”
- *Fråga 11* -”Det är lättare att söka efter relevant information. Fria kommentarer kan bli lite väl koncisa ibland.”
- *Fråga 12* - ”Nej absolut inte, förmodligen en av de sämsta idéer jag någonsin testat”
- *Fråga 13* - ” ja. Omotiverande, tycker inte de fyller någon funktion.”
- *Fråga 14* - ”Jag såg inte riktigt fördelen med det. Antingen så ska man ju implementera en story och då finns det inte så mycket att skriva förutom det som står på storykortet. Ska man fixa bug eller refaktorisera så vet man ju inte vad som ska göras i förväg.”, ”I detta relativt lilla projekt kändes det mest överdrivet att sitta och skriva kommentarerna innan man gjort ändringarna. Men jag antar att det kan bli bra bara man tvingar sig att göra det några gånger.”, Det är inte alltid man vet hur man ska lösa ett problem förrän man har byggt upp strukturen omkring det. Därför kan det vara irriterande när det tar tid (som kunde använts till att knacka kod) att formulera vad man kommer göra. Jag vet iofs inte om jag kan kalla det ett problem, snarare ett irritationsmoment.”
- *Fråga 17* - ” Nej, hade förmodligen utformat egna liknande mallar sedan haft specifikationer för hur själva skrivandet skulle se ut”

# 6 Slutsatser

Till att börja med är det värt att nämna att vi har testat detta på ett begränsat antal grupper och slutsatserna är därför baserat på ett ganska tunt underlag. Kvaliteten på commit comments varierade även en del mellan de olika grupperna. Dessutom kan studien anses vara subjektiv, då det det är vi som definierat hur en bra respektive en dålig commit-kommentar ser ut.

Resultatet var bättre än vi trodde ur den aspekten att antalet helt tomma kommentarer var betydligt färre än väntat. Man fick till och med leta ett tag för att hitta exempel på dem. Det samma gäller värdelösa kommentarer i stil med de exempel listade i resultatdelen av rapporten. En förklaring till detta kan vara att man känner sig obligerad att skriva något när rutan finns och man har gjort ändringar. Det känns som en bra idé, och inom ”software engineering” är det klassat som en god praktik, att alltid skriva en liten kommentar om man ändå commitar kod in till repositoriet. En annan anledning till att resultaten har varit över förväntan kan vara att under EDA260 har studenterna fått en introduktion till SCM[2] och har även haft en laboration med CVS. [4].

Även om antalet tomma och värdelösa commitkommentarer var färre än vi trodde, så hade många kommentarer vad vi kallar otillräckligt med information. Det vill säga att kommentaren har någon information av värde men uppfyller inte kraven för en ok eller bra commit comment. Att det skulle vara

[2]Software Configuration Management

relativt få commit comments som kvalificerade sig som ok eller bra, är ekvivalent med vår första hypotes. Den blev också bekräftat i studien av commit comments utan mallar, då antalet värdelösa (12 %) och otillräckliga (47 %) var totalt 59 % under första iteration och antalet bra kommentarer bara var 2 %. Även på de tidigare årets repositorier var kommentarer generellt sätt dåliga då antalet värdelösa var 20 % och otillräckliga var 50 % och där bara var en enda kommentar som klassades som bra utav 179 stycken kommentarer. Resultatet var var alltså i enlighet med vår hypotes.

Under iteration 2 lät vi utvecklarna i de tre grupperna använda våra mallar. Resultatet var slående. Antalet bra kommentarer denna iteration hade ökat till 45 % och värdelösa och otillräckliga kommentarer var nere på 33 %.. Detta gör att vi, baserat på vårt resultat, kan dra slutsatsen att man genom något så enkelt som mallar kan få utvecklarna att höja kvaliteten på sina commit comments avsevärt.

Undersökningen vi gjorde på 2007 års repositorier visar en stark indikationen på att teamens commit comments blev sämre med tiden. Värdelösa kommentarer ökade med hela 37 procentenheter samtidigt som andelen ok commit comments minskade med 17 procentenheter. Detta beror förmodligen på att utvecklarnas projekt ingår i en obligatorisk kurs, i vilken deltagarna efter hand tappar motivationen. Dessutom lär folk känna varandra bättre inom teamet, vilket får antalet skämtsamma kommentarer att öka. Detta i enlighet med vår fjärde hypotes.

XP-Projektet i kurs EDA260 är ett ganska litet projekt och detta begränsar vikten av commit comments. Enligt enkäten har majoriteten av utvecklarna har inte tittat på några gamla commit comments, då de hellre har frågat teamet om ändringar som gjorts. Eftersom det är ett litet projekt som sträcker sig över en kort tidsperiod, kan utvecklarna dels sitta nära varandra och dels komma ihåg de flesta ändringar som gjorts. Detta är dock inte möjligt i samma utsträckning på större projekt vilket förmodligen var anledningen till att 100% av utvecklarna som svarade på enkäten tror de kommer haft mer användning av commit comments i större projekt. Mallar för commit comments bör därför anpassas efter hur stort projektet är som de ska användas i.

Mallarna som användes är utformade för att ge tillräckligt med information samtidigt som de skulle underlätta för utvecklarna att skriva. Baserat på resultaten från enkäten, så var utvecklarna överlag mycket nöjda med mallarna. Det var ingen informations som saknades i dem, vettiga kommentarer gick fortare att skriva och hela 78 % tyckte att den bästa metoden var att använda mallar efter implementeringen. Slutsatsen man kan dra från det är mallarna uppskattades och faktiskt var lätta och bra att använda. Vi noterade dock att utvecklarna undvek att använda mallarna vid stress och tidsbrist exempelvis vid releaser.

Vi kände att alla mallarna användes och ingen av dem var onödiga. Den helt klart mest använda mallen var den för story/task implementerad, eftersom detta var den största orsaken till att en commit gjordes. Det fanns dock även många bra commit comments där mallar för buggfixning och refaktorisering hade använts. Vi fick inte heller någon kommentar på att det var någon typ av mall som saknades. Dock så fanns det fall då ingen mall riktigt passade, till exempel då bara testfall eller javadoc skrivits. Då valde utvecklarna ofta istället att inte använda någon mall.

*Commit comment first* verkade inte alls gå hem bland utvecklarna. De flesta såg ingen mening med det och tyckte inte det bidrog med något. Många såg det som ett irritationsmoment och något som de gärna hade skippat. Bara 3 stycken av 18 tillfrågade sa att de fick hjälp av metoden och bara 4 stycken tyckte det var genomförbart. Problemen med det var att de ofta glömde bort när de fick en story att de skulle skriva *commit comment first*. Andra problemet var att det ofta är svårt att veta exakt var ändring ska ske och att det aldrig blir som man tänkt sig från början. Eftersom det inte hjälpte vid varken design eller taskindelning, så drog vi slutsatsen att idén inte är särskilt användbar. Utifall vi gett utvecklarna en bättre introduktion till*commit comment first* och det skulle testas under en längre tidsperiod skulle dock eventuellt resultatet bli annorlunda.

Att på ett standariserat sätt skriva mallar, har enligt vår studie visat sig vara användbart. Det gör att utvecklarna får med all viktig information vid en commit comment, så att nästa utvecklare får en bra överblick av vad som har ändrats. Vidare går det både snabbare och enklare och blir bättre kvalitet på

commit-kommentarerna när mallarna tillämpas. Mallar är alltså ett bra verktyg för att förbättra kvaliteten och göra commit comments användbara för tillbakablickar, spårning och sökning i koden, medan *commit comment first* är en metodik som inte bidrog med speciellt mycket och var svårtillämpad.

# 7 Referenser


1. Annie Chen, Eric Chou, Joshua Wong, Andrew Y. Yao, Qing Zhang, Shao Zhang, Amir Michail, *CVSSearch: Searching through Source Code using CVS Comments* icsm, p. 364, 17th IEEE International Conference on Software Maintenance (ICSM'01), 2001

2. Andersson, J., Göransson, D., Jakobsen, A., Jönsson, A. *Possible approaches to a traceable CM system*, 2007, EDA240 Konfigurationshantering. Department of Computer Science, Lund Institute of Technology.

3. Zipfel, T., Månsson, M., Nygren, S. *The need of traceability in the software development process.* 2007, EDA240 Konfigurationshantering. Department of Computer Science, Lund Institute of Technology.

4. Geraldine Fitzpatrick, Paul Marshall,Anthony Phillips.*CVS integration with notification and chat: lightweight software team collaboration.* Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work p.49-58, 2006.

5. Repositories from the course EDA260 - Programvaruutveckling i grupp, 2007 and 2008.

Appendix A

# Mallar

## Implementering

Story/Task implementerad:
Implementationer och Ändringar gjorda i klasser:
Utvecklare:

## Buggfixning

Kort beskrivning av buggen:
Kort beskrivning av fixen:
Ändringar har gjorts i följande klasser/metoder:
Utvecklare:

## Refaktorisering

Typ av refaktorisering:
Kort beskrivning av Ändringarna:
Ändringar har gjorts i följande klasser:
Utvecklare:

# Enkät/Utvärderingsenkät

1. Har du någon gång läst en gammal commit comment? (Inte enbart för nöjes skull)
2. Har du behövt fråga någon i teamet om gamla ändringar någon gång?
3. Tror du att commit comments faktiskt hade hjälpt ifall man hade glömt bort gamla ändringar?
4. Tror du att du hade haft mer nytta av commit comments i ett större projekt?
5. .Har mallarna underlättat att skriva vettiga commit comments? Om ja, på vilket sätt?
6. Har det varit jobbigt att behöva använda mallarna? Om ja, varför?
7. Går det fortare att skriva commit comments med en mall?
8. Har mallarna fått dig att anstränga dig mer för att tänka igenom det du skriver i din commit comment?
9. Har du skrivit bättre commit comments med mallarna? Om, ja på vilket sätt?
10. Känner du att du skulle behöva mer stöd för att kunna skriva bättre commit comments?Om ja, hur?
11. Är det bra med en generell standard för commit comments eller är det bättre att få skriva fritt? Motivera
12. Har det hjälpt dig på något sätt vid implementeringen att utforma commit comments först?Om ja, på vilket sätt?
13. Var det jobbigt att skriva commit comments före implementeringen? Om ja, varför?
14. Var idén att skriva commit comments först genomförbar eller ställde den mest till problem? Vilka problem isåfall?

15. Fattades något i mallarna, och isåfall vad?

16. Prioritera vilket av följande du ansåg var det bästa sättet att skriva commit comments på: Utan mallar (UM), med mallar efter implementering(EI),med mallar före implementering(FI) i kategorierna bäst, nästbäst och sämst.

17. Skulle du vilja använda mallarna för commit comments i framtida projekt? Motivera

| Fråga | Ja[st] | Nej[st] | Vet ej[st] |
|---|---|---|---|
| 1 | 4 | 14 | 0 |
| 2 | 12 | 6 | 0 |
| 3 | 15 | 3 | 0 |
| 4 | 18 | 0 | 0 |
| 5 | 14 | 3 | 1 |
| 6 | 3 | 15 | 0 |
| 7 | 15 | 3 | 0 |
| 8 | 7 | 10 | 1 |
| 9 | 14 | 4 | 0 |
| 10 | 2 | 16 | 0 |
| 11 | 14 | 1 | 3 |
| 12 | 3 | 13 | 2 |
| 13 | 7 | 6 | 5 |
| 14 | 4 | 9 | 5 |
| 15 | 0 | 14 | 4 |
| 17 | 7 | 3 | 10 |

| Fråga | Ja[%] | Nej[%] | Vet ej[%] |
|---|---|---|---|
| 1 | 0,22 | 0,78 | 0 |
| 2 | 0,67 | 0,33 | 0 |
| 3 | 0,83 | 0,17 | 0 |
| 4 | 1 | 0 | 0 |
| 5 | 0,78 | 0,17 | 0,06 |
| 6 | 0,17 | 0,83 | 0 |
| 7 | 0,83 | 0,17 | 0 |
| 8 | 0,39 | 0,56 | 0,06 |
| 9 | 0,78 | 0,22 | 0 |
| 10 | 0,11 | 0,89 | 0 |
| 11 | 0,78 | 0,06 | 0,17 |
| 12 | 0,17 | 0,72 | 0,11 |
| 13 | 0,39 | 0,33 | 0,28 |
| 14 | 0,22 | 0,5 | 0,28 |
| 15 | 0 | 0,78 | 0,22 |
| 17 | 0,39 | 0,17 | 0,56 |

| Omdöme | Utan mallar[st] | Mallar först[st] | Mallar sist[st] |
|---|---|---|---|
| Bäst | 1 | 3 | 14 |
| Nästbäst | 11 | 3 | 4 |
| Sämst | 6 | 12 | 0 |

| Omdöme | Utan mallar[%] | Mallar först[%] | Mallar sist[%] |
|---|---|---|---|
| Bäst | 0,06 | 0,17 | 0,78 |
| Nästbäst | 0,61 | 0,17 | 0,22 |
| Sämst | 0,33 | 0,67 | 0 |

Figur 2: Enkätsvar