Utvärdering av parprogrammering

IV1303 Modern Mjukvaruutveckling

Dana Ismail, Mehir Wolde
Skolan för elektroteknik och datavetenskap (EECS)
Kungliga Tekniska Högskolan
(KTH) Sverige
dismail@kth.se , mswolde@kth.se

Abstract

In modern society, software and the development of it has become very important in many types of environments. It has become very common to apply different forms of methods in the hopes of improving efficiency, precision and correctness.

This research is about the effectiveness and actual usefulness of pair programming. Is pair programming a good tool to achieve these goals mentioned above? The study is based on work in software-related projects where this method is applied. The goal is to investigate and evaluate the advantages and disadvantages of pair programming and to examine the workers' willingness to use this method.

The study is based on five evaluation criterias - accuracy, synchronization, driver, observer and efficiency. Results from this research were positive. It was shown that developers with less experience benefit significantly more from pair programming than experienced programmers.

Keywords

Software development, working methods, project, software, pair programming, practices.

Innehållsförteckning

1. Inledning

Inom branschen för mjukvaruutveckling finns det ett flertal olika arbetssätt och metoder att tillämpa i sina diverse sorters av arbete. Dessa metoder har som syfte att effektivisera, underlätta samt att göra arbetet behagligt för den som utför den. Några exempel på sådana tillämpningar för olika sysselsättningar inom mjukvaruutvecklingen är parprogrammering, rotation, test-driven utveckling, kontinuerlig integration, granskning av kod, refactoring, scrum etc. Dessa exempel på metoder förekommer vanligen i projektarbeten inom företag som tillhandahåller produkter baserat på mjukvara.

Denna rapport har som syfte att behandla ämnet och metoden parprogrammering. Parprogrammering är en väldigt frekvent förekommande metod som används av ett större antal etablissemang världen över. Just parprogrammering förekommer endast inom företag som inriktar sig i mjukvara där utvecklarna faktiskt genererar kod genom att programmera i par. Dock återfinns denna modell i olika varianter. Företag som inte erbjuder mjukvara kan ha liknande metoder. Arbetare i dessa företag behöver inte specifikt skriva kod tillsammans dock kan de tillämpa ett liknande arbetssätt där de arbetar i par för att framställa en viss komponent exempelvis.

Det finns mycket skilda åsikter angående parprogrammering. Vissa anser att det är en bra tillämpning och andra är emot denna typen av arbete. Denna modell verkar uppskattas av många samtidigt som vissa anser att det finns viss problematik med den.[1]

I denna rapport ska det påvisas hur effektivt parprogrammering egentligen är. Undersökningen ska baseras på arbete inom mjukvarurelaterade projekt där denna metod är tillämpad. Målet är att undersöka och utvärdera fördelar och nackdelar med parprogrammering samt att undersöka arbetarnas vilja till att använda sig av denna metod. Rapporten följer denna struktur: avsnitt 2 redovisar en utvidgad bakgrund om ämnet, avsnitt 3 redovisar forskningsmetoden, avsnitt 4 visar undersökningens resultat, i avsnitt 5 finns en analys och diskussion, avsnitt 6 redovisar en slutsats samt diskussion och till slut avsnitt 7 redovisar referenser.

[1] https://ieeexplore-ieee-org.focus.lib.kth.se/document/7274252, W. Sun, G. Marakas, M. Aguirre-Urreta, IEEE, 2016,

2. Utvidgad bakgrund

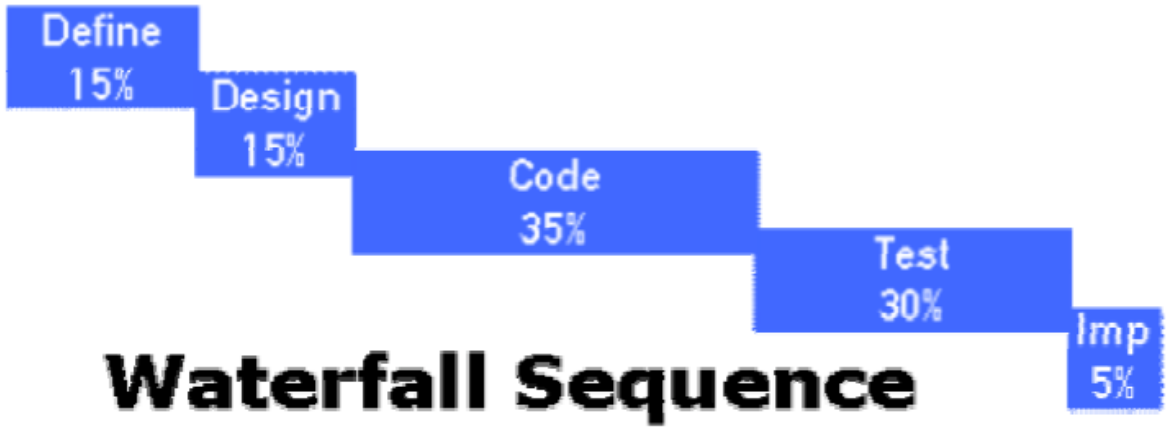
Denna sektion av rapporten kommer att innehålla en teoretisk bakgrund med information angående traditionella och agila metoder i avsnitt 2.1. Detta kommer att innehålla information om agila samt traditionella metoder för systemutveckling. Sedan beskrivs den metod som rapporten utvärderar och det projekt som utförts i samband med rapporten.

2.1 Traditionell utvecklingsmetod

Den traditionella metoden riktar sig mot att utföra ett strukturerat arbete som följer ett sekventiellt flöde av bestämda faser som måste avslutas innan en nästföljande fas kan påbörjas. Faserna kan delas in i 5 områden: *vad* systemet kommer att utföra, *hur* systemet kommer att utformas, *kod* utveckling, *testfas* samt *implementation*. Detta flöde av avslutade faser kan visualiseras som ett vattenfall vilket är varför metoden även benämnts som vattenfallsmodellen.[2]

2.2 Agil mjukvaruutveckling

Den agila metoden skiftar sig ifrån den traditionella metoden genom att repetera att flertal dynamiska cykler. Dessa cykler förser arbetet med kontinuerlig inlärning samt möjligheter att anpassa sig efter projektets nuvarande tillstånd. Agil utveckling är mer utformad efter verkligheten eftersom att den erkänner behovet av att återvända och modifiera tidigare arbetsuppgifter. Denna metod är även kallad för iterativ eftersom att det till stor del krävs att uppgifter iterativt modifieras ett flertal gånger innan en produkt eller ett projekt har blivit färdigställt.[3]



Figur 2.1: **visualisering av vattenfallsmodellen**[4]

[2] https://www.academia.edu/download/58993716/10.1.1.464.609020190422-13963-j0ju8a.pdf, M. A. Awad, publicerad i Academia 2005
[3] https://ieeexplore.ieee.org/abstract/document/6480417, A. Aitken and V. Ilango, publicerad i IEEE 2013
[4] D. Marks, "Development Methodologies Compared", N CYCLES software solutions, December 2002 , www.ncycles.com

2.3 Jämförelse av traditionell- och agil mjukvaruutveckling

Traditionell mjukvaruutveckling introducerades först under 1970-talet och har genomgått en stor utveckling sedan dess tillkomst. Metoden har haft stor framgång men trots detta har den även ett flertal nackdelar jämfört med agil utveckling. Eftersom att den traditionella metoden arbetar linjärt skapar det få möjligheter att anpassa projektet till förändringar i efterhand. Detta är en stor fördel inom den agila metoden som hanterar instabila krav som kan förändras under projektets cykel.

Agila metoder används till stor del inom mindre projekt. Om projektet är av större skala krävs det ett flertal resurser, fler arbetar samt bättre organisering. Dessa omständigheter leder till att fler krav skapas som gör att projektet kräver en förbättrad struktur. Detta är något som kan uppnås genom bruket av traditionella metoder eftersom att dessa dessa tillhandahåller planer och processer för bättre kommunikation.

2.4 Mjukvaruutvecklingspraxis

Det finns ett stort antal olika mjukvaruutvecklingspraxis. Dessa tillämpas på olika sätt och har diverse syfte samt ändamål. En mjukvaruutvecklingspraxis är någon form av metod eller process som ska stödja, förbättra eller effektivisera ett arbete inom den moderna mjukvaruindustrin. Detta arbetssätt ska resultera i positiv påverkan på arbetarnas kompetenser samt att förbättrad arbetsmiljö och kommunikation i vissa fall.

En lista på välkända mjukvaruutvecklingspraxis kan återfinnas i listan nedan (elementen i listan är slumpmässigt valda, ingen hänsyn till ordning):

- Refaktorisering
- Riskhantering
- Iterativ utveckling
- Informativa arbetsmiljöer
- Planering
- Tidsuppskattning
- Kommunikation
- Retrospektiv

Dessa tillämpningar påstås vara kärnan i den moderna mjukvaru-industrins arbetssätt. Därutöver kan resultat skifta på olika arbetsplatser beroende på hur det dagliga arbetet för respektive utövare eller arbetare ser ut.[5]

[5] https://dl.acm.org/doi/pdf/10.1145/3368089.3409766, Yvonne Dittrich et al, publicerad i ACM Digital Library 2020

2.5 Parprogrammering

Parprogrammering är en effektiv teknik inom modern mjukvaruutveckling. Det är en typ av programmering som utför av två programmerare som arbetar gemensamt. Detta par separeras in i två växlande roller som efter en bestämd tid skiftas för att ge båda parterna en möjlighet att utföra båda rollerna. De två rollerna kallas för *förare* samt *observatör*/*navigatör*. Föraren utför det praktiska arbetet, detta kan variera mellan att skriva kod, utföra tester alternativt utveckla algoritmer. Observatören utför ett mer teoretiskt arbete genom att uppmärksamt identifiera defekter i förarens arbete. Dessa defekter kan skifta mellan strategiska och taktiska defekter. Taktiska defekter kan variera mellan syntaxfel, skrivfel eller att tillkalla fel metod. Strategiska defekter sker när förarens implementering av programmet inte kommer att uppfylla målet med projektet. Eftersom att observatören inte är involverad i det praktiska arbetet skapar det en ytterligare synvinkel på programmet som kan inkludera en mer opartisk åsikt.[6]

Metoden har genom dess utveckling sett stora skillnader och ett flertal nya tillvägagångssätt har introducerats. Med hjälp av ny teknik och nya innovationer har man nu möjligheten att utföra denna metod på separata platser, även kallad för distribuerad parprogrammering. Skärmdelning med hjälp av olika online applikationer gör det möjligt för navigatören att observera förarens arbete utan att arbeta på samma dator. Existerande integrerade utvecklingsmiljöer har utvecklat extensioner som tillåter samarbete mellan två datorer i samma dokument vilket skapar möjlighet för navigatören att notera problem i koden samtidigt som föraren arbetar.

[6] https://collaboration.csc.ncsu.edu/laurie/Papers/ESE%20WilliamsPairProgramming_V2.pdf , Laurie Williams, publicerad vid CSU NCSU 2003

2.6 Våra projekt

Denna rapport är delvis baserat på två olika projekt som författarna ägnat sig åt. Resultatet blir att denna framställningens utgångspunkt och primära källa är dessa två projekt. Dessa två avgränsade arbeten har olika produktresultat dock ska arbetssättet för arbetarna för respektive projekt vara identiskt där man använder samma metoder och tillämpningar bland utvecklarna. Detta leder till en positiv fördel eftersom att undersökningen blir mer exakt och sedd ur olika perspektiv. Det är fortfarande samma teknik som används trots olika slutprodukter.

Första projektet handlar om att framställa en hemsida/app som ska agera som ett beslutstödssystem för civilbefolkningen. Produktens huvudsakliga syfte handlar om att underlätta dem val en person tar under en dag. Till exempel kan det på hemsida visas information om vädret för en viss region från flera olika källor.

Andra projektet handlar om att framställa en hemsida/app som erbjuder ett system för arbetsgivare och arbetstagare att lättare kunna kommunicera, tilldela arbeten samt att registrera olika händelser. Till exempel kan en admin tilldela en arbetstagare en specifik uppgift genom denna hemsida.

Båda dessa arbeten görs på liknande. Eftersom utvecklarna i grupperna för respektive projekt kommer från olika bakgrunder gällande arbetssätt så bestämdes det inom båda grupper att parprogrammering är en central del i arbetet för att få alla arbetare på samma bana. Inom båda projekten tillämpades metoden Scrum vilket är en praxis i sig. Parprogrammeringen ansågs vara en bra metod inom varje projekt eftersom att arbetarna skulle dela med sig av sina kunskaper medans projektets arbete gick framåt.

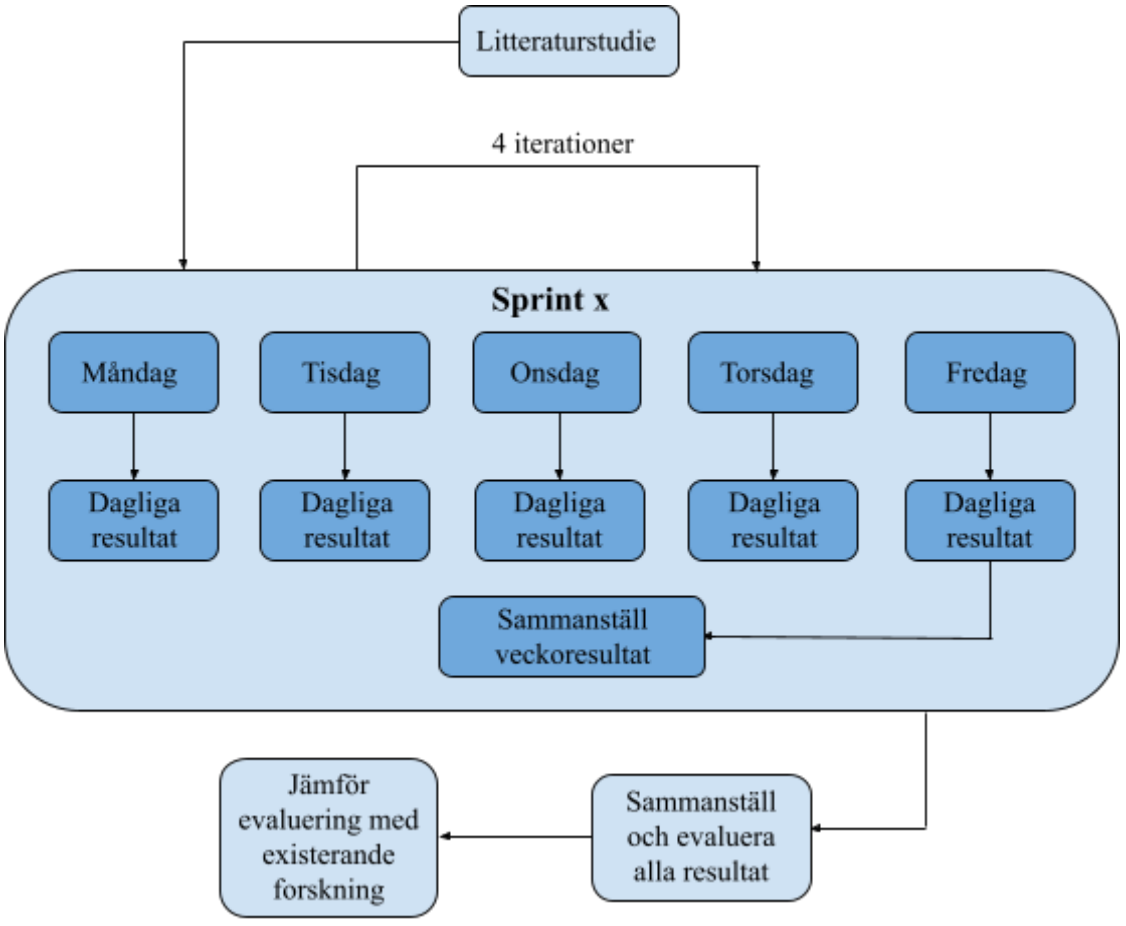
3. Metod

Denna del av rapporten kommer att beskriva den forskningsmetod som använts. Kapitel 3.1 återger de olika forskningsfaser som utförts under projektet. Kapitel 3.2 redogör för utvärderingsmodellen.

3.1 Forskningsfaser

Under projektets förlopp genomfördes ett forskningsarbete som bestod fyra komponenter. Dessa komponenter illustreras i figur 2. De fyra olika komponenterna som framgår i figuren beskrivs som följande:

Litteraturstudie:
En litteraturstudie har utförts för att erhålla en betryggande mängd kunskap angående den valda metoden som rapporten ska utvärdera. Denna del av forskningsarbetet förser även rapporten med gynnsam information inför avsnittet som beskriver bakgrunden till den valda metoden. Rapporter från varierande databaser utgjorde källorna för litteraturstudien och är även inkluderade som källor för denna rapport.



Figur 3.1: Illustrering av de olika stegen under forskningsarbetet

Analys under genomförande av projekt:
Den andra komponenten av forskningsarbetet bestod av att samla in data under projektets gång. Projektet utfördes under 4 sprints, varav varje sprint varade under fem arbetsdagar. Efter varje arbetsdag analyserades det som utförts samt antecknades för att användas som data till rapporten. Vid varje sprints avslut sammanställdes ett gemensamt veckovist resultat som sedan skulle jämföras med framtida anmärkningar. Denna data utvärderas och bygger på de olika kriterierna som framgår i kapitel 3.2 av denna rapport.

Sammanställning/evaluering av samtliga resultat:
Vid projektets avslutning sammanställdes ett slutgiltigt resultat utifrån de olika resultaten som kommit fram från den föregående fasen. Detta slutgiltiga resultat bestämdes genom att evaluera de olika resultaten från slutet av varje vecka utifrån de olika kriterier som valts inom utvärderingsmodellen.

Jämförelse med existerande studier:
Efter att resultatet sammanställs blev det bedömt samt jämfört med existerande studier, rapporter och vetenskapliga artiklar inom det valda området.

3.2 Utvärderingsmodellen

För att kunna utvärdera parprogrammering effektivt och med precision krävs det att man har specifika och konkreta kriterier som ska efterföljas. Detta beror på att tankarna kring hur pass bra eller dåligt ett ämne som till exempel parprogrammering är beror på vem man frågar. Med andra ord är uppfattningen om lönsamheten av parprogrammering väldigt subjektivt. Därav krävs det att man specificerar olika utvärderingskrav som ska täcka många synvinklar kring detta ämne. För denna undersökning fanns följande kriterier för utvärdering:

- **Korrekthet:** I denna undersökning var en av utvärderingskriterierna att kontrollera hur parprogrammering påverkar hur pass korrekt eller felfritt ett stycke mjukvara blir. Detta kontrollerades genom att en mindre undersökning skedde efter varje session av parprogrammering. Det som granskades var *antalet fel* som upptäcktes efter att man ansett att uppgiften var färdig.

- **Synkning:** Ett annat utvärderingskriterium var att undersöka hur synkningen mellan dem två programmerarna påverkade arbetet. Om de två personerna kommer från olika bakgrund gällande nivå av programmeringserfarenhet krävdes det en viss synkning mellan dessa. Detta kunde ta lite tid samt viss diskussion. Frågeställningen var att undersöka efter varje synkning hur det kändes för utvecklarna och om det ledde till nya ideer kring hur man skulle arbeta framöver med varandra med tanke på likheter och skillnader mellan personerna.

- **Förare:** Detta kriterium utgick ifrån att evaluera hur personen som agerade som föraren upplevde utförandet av parprogrammering utifrån förarens perspektiv jämfört med hur vanlig programmering upplevs. Syftet med detta kriterium var att evaluera beteendet av hur en programmerare utför ett arbete i en roll som sedan skulle evalueras av en annan person.

- **Observatör/Navigatör:** Detta kriterium utgick ifrån att evaluera hur personen som agerade som observatören upplevde utförandet av parprogrammering utifrån observatörens perspektiv jämfört med hur vanlig programmering upplevs. Syftet var att evaluera hur en mer teoretiskt syn på programmering kan upplevas samt dess för- och nackdelar.

- **Effektivitet/Lönsamhet:** Det sista kriteriet handlar om hur pass effektivt eller lönsamt det var att arbeta med parprogrammering som verktyg. Detta undersöktes med hjälp av att mäta tiden det tog för att utföra vissa uppgifter. Inför varje sprint fanns det en sprintplanering där medlemmar i gruppen satte upp uppgifter som skulle göras under kommande vecka. Dessa uppgifter uppskattades i förväg hur lång tid de skulle ta med hjälp av bland annat planning poker. För denna undersökning mäter vi tiden för hur lång tid det tog att genomföra den jämfört med den uppskattade tiden.

4. Resultat

I detta kapitel redovisas resultaten för undersökningen av parprogrammering. Detta kapitel följer ordningen av utvärderingskriterier som återfinns under kapitel 3.2. Under kapitlen 4.1 - 4.5 beskrivs varje utvärderingskriterium var för sig.

4.1 Korrekthet

Det undersöktes om hur pass korrekt och felfritt ett stycke mjukvara blev i slutet av en session där parprogrammering har använts som arbetsmetod. Efter att ha arbetet med parprogrammering två till tre gånger per vecka i fyra veckors tid har det utvecklats en bra uppfattning kring ämnet. Korrektheten kontrollerades i form av antal upptäckta fel efter varje session av parprogrammering.

Resultaten för detta kriterium kan visas i form av en tabell där antalet fel per session är inskrivet. Tabellen visas i slutet på denna sida (Figur 4.1).

Som det kan observeras i tabellen nedan så visar studien att antalet upptäckta felaktigheter per sittning inte passerat tre stycken fel per session. Det kan dessutom räknas ut ett medelvärde för dessa fyra veckor. Medelvärdet för antal fel per session blir då 1,7 (17 antal fel totalt / 10 sittningar).

	Vecka 1	**Vecka 2**	**Vecka 3**	**Vecka 4**
Session 1	3	2	1	2
Session 2	1	1	2	1
Session 3	2	-	-	2

Figur 4.1: Denna tabell visar antalet upptäckta fel per session och vecka.

4.2 Synkning

I avsnitt 3.2 beskrivs det om hur detta kriterium evaluerats utifrån hur deltagarna upplevt arbetet samt hur denna upplevelse påverkat deras tankesätt inför framtida arbeten.

En stor likhet i de två olika projekten som utförts var deltagarnas tidigare kunskap angående de valda programmeringsmiljöer samt språket. I början av projektet utfördes en undersökning för att se hur de olika deltagarna bedömde sina programmeringskunskaper. Resultaten från denna undersökning gav att majoriteten av deltagarna bedömde sig lägre ifall de inte kom från en mer teknisk bakgrund medans det gav det motsatta för de som kom från en mer teknisk bakgrund. Med hjälp av denna betygsskala blev paren indelade i både folk som hade

likvärdig kunskap samt i par vars kunskapsnivåer skiljde sig. Denna indelning gav olika resultat angående synkning för de olika paren.

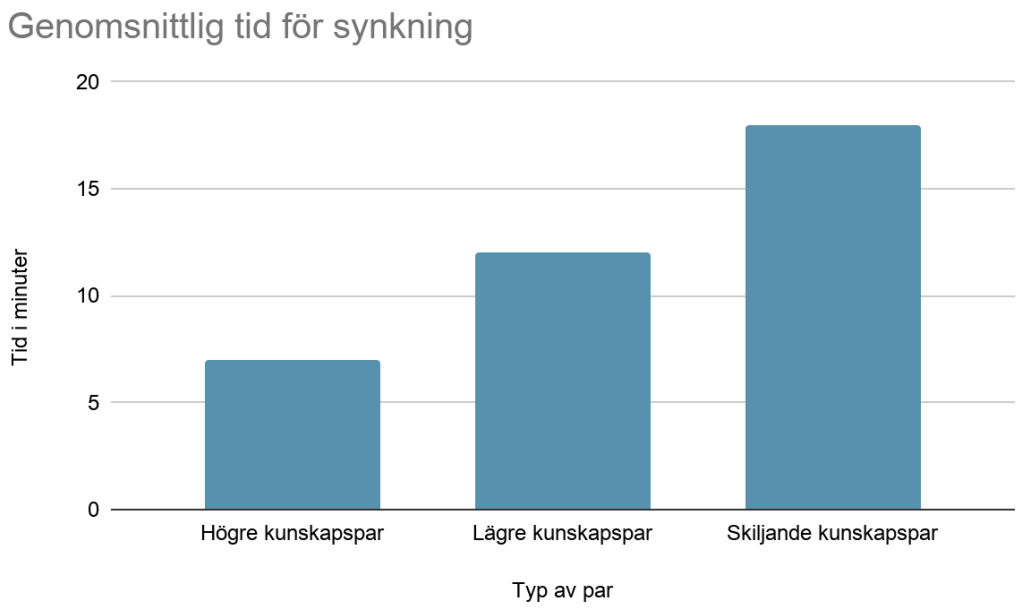


Diagram 4.2: Kolumndiagram som visar den genomsnittliga tiden för synkning i de olika sorters paren

Från diagrammet ovan kan man vittna att det finns en proportionell skillnad i hur tiden för att synkas ökas i och med att kunskapsnivån skiljer sig mer. Paren som uppfattades behärska högre kunskaper inom programmering krävde en mindre tid, medelvärde på sju minuter, för att utföra sin synkning. Paren som uppfattades behärska lägre kunskaper krävde mer tid än de med högre, genomsnittlig tid på 12 minuter. Det par som bestod av deltagare med skiljande kunskapsnivåer krävde den längsta tiden för synkning med ett medelvärde på 18 minuter. Resultaten gav även att tiden för denna typ av par samt par med högre kunskapsnivåer förändrades på en minimal skala medans tiden för par av lägre nivåer såg en större förminskning under projektets gång.

4.3 Förare

	Vecka 1	Vecka 2	Vecka 3	Vecka 4
Session 1	9	7	10	8
Session 2	10	9	7	9
Session 3	8	-	-	9

Figur 4.3: Tabell som visar förarens betygsättning, 1-10.

Det har dessutom examinerats om hur föraren har ansett att det har gått och förarens erfarenhet angående parprogrammering efter dessa fyra veckor av arbete där parprogrammering har implementerats till arbetssättet. Granskningen har gått ut på att föraren har fått beskriva deltagande/engagemang och hur det har gått efter varje session. Beskrivningen gick ut på att föraren fick ange ett nummer från noll till tio, där noll var väldigt dåligt och tio var väldigt bra, efter varje sittning. Detta resultat kan visas i form av en tabell som återfinns i början på denna sida (Figur 4.3).

Det lägsta värde som en förare angivit underskrider inte siffran sju. Föraren har angivit siffror mellan sju och tio. Ett medelvärde kan räknas ut med tanke på resultatet i tabellen ovan. Det har varit tio stycken sittningar där parprogrammering har skett enligt tabellen. Ett medelvärde blir då 8,6 (86 i betygsättning totalt / 10 sittningar).

4.4 Observatör/Navigatör

Likt kriteriet angående föraren, evalueras detta kriterium på ett överensstämmande sätt. Deltagaren fick beskriva sin upplevelse utifrån observatörens perspektiv samt hur mycket personen kände att den deltog i arbetet på en skala från noll till tio.

Utifrån tabellen nedan (tabell 4.4) kan det observeras att observatören ej bedömdes på ett värde som översteg 9 samt att medelvärdet 7.2. Första veckans sessioner gav ett betydande mindre värde jämfört med sista veckan i slutet av projektet. Detta samt andra resultat kommer att analyseras vidare i kapitel 5.

	Vecka 1	Vecka 2	Vecka 3	Vecka 4
Session 1	5	7	9	8
Session 2	6	8	9	9
Session 3	6	-	-	8

Tabell 4.4: Tabell som visar den genomsnittliga bedömningen för förare

4.5 Effektivitet/Lönsamhet

Det sista kriteriet som undersöktes var effektiviteten och lönsamheten när man har arbetet med parprogrammering. Detta kriterium granskades genom att jämföra tiden det tog för att utföra en specifik uppgift med en uppskattning om hur lång tid det skulle ta. Denna uppskattning gjordes i början av veckan tillsammans med teamet.

För denna granskning har det gjorts ett kolumndiagram där det påvisas ett medelvärde för de fyra veckorna. Diagrammet kan återfinnas nedan (Figur 4.5).

Som det kan observeras i diagrammet nedan så har antalet timmar för att slutföra arbeten i genomsnitt varit lägre än antalet timmar som man har estimerat. Med andra ord har arbetet genom parprogrammering lett till att uppgifter blir klara innan den uppskattade tiden för respektive uppgift.

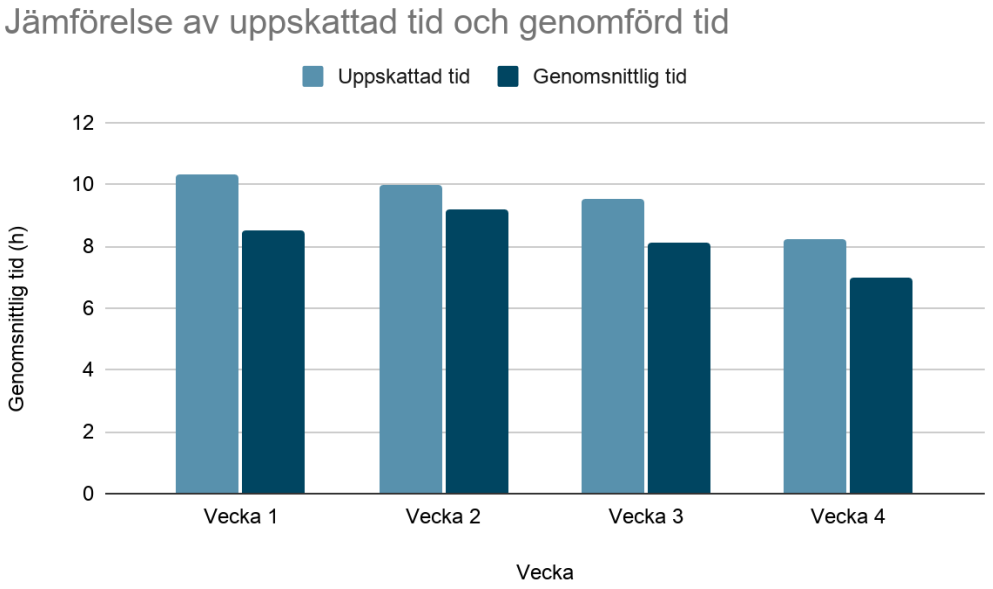


Diagram 4.5: Detta diagram visar genomsnittlig antal timmar som är estimerade gentemot verkliga timmar det tagit att utföra arbetet.

5. Analys och Diskussion

I denna del av rapporten analyseras samt diskuteras resultaten från denna undersökning. Denna utvärdering ska besvara frågeställningen som återfinns i inledningen av denna rapport. Analysering sker ur ett perspektiv som ska symbolisera utvärderingskriterierna som finns under *3.2 Utvärderingsmodellen*.

5.1 Korrekthet

Meningen med att undersöka detta utvärderingskriterium var att kontrollera hur parprogrammering påverkar hur pass korrekt eller felfritt ett stycke mjukvara blir. Detta granskades genom att mindre kontroller genomfördes efter varje session av parprogrammering. Det som rannsakades var antalet fel som upptäcktes efter att man ansett att uppgiften var färdig.

Som det kan observeras i resultatet för detta kriterium under rubriken *4.1 Korrekthet* så har antalet upptäckta felaktigheter per sittning inte passerat tre stycken fel per session. Det skedde totalt tio sittningar där parprogrammering användes som verktyg. Under dessa sittningar upptäcktes totalt 17 stycken fel. Detta resulterar i ett medelvärde på 1,7 fel per sittning, vilket är ett väldigt bra resultat.

Denna undersökningens utfall är bra eftersom resultatet i denna undersökning står i likhet med ett verk från ett experiment som omnämns i en journal utgiven av IEEE. I detta experiment har man undersökt hur bland annat korrektheten bland "junior" utvecklare påverkas av parprogrammering. Då påvisades att korrektheten bland dessa utvecklare i allmänhet ökade med 73 procent.[7] Intermediära programmerare som parprogrammera ökade korrektheten med 4 procent samt dessvärre för "senior" utvecklare minskades korrektheten med 8 procent enligt samma journal. I en annan journal skriven av Matthias M. Müller och publicerat av ScienceDirect så har en liknande undersökning gjorts där utvecklarna som utförde olika tester där parprogrammering hade stor betydelse uppnådde man en nivå av korrekthet som är ungefär 95 procent.[8]

Dessa undersökningar fastslår tillsammans att parprogrammering är ett väldigt effektivt verktyg för att uppnå minimal felaktighet. Detta gäller främst för utvecklare mindre till intermediär erfarenhet av programmering samt arbeten i projekt. I denna forskning har vi författare arbetet i projekt där medlemmarna kategoriseras som nybörjare till det programmeringsspråk som används under projektet. Detta är en anledning till varför dessa bra resultat uppnåddes. Som medlem med mindre erfarenhet känns det att man lärde sig mycket mer än om man skulle programmerat själv, vilket stärks med bevisningen från denna

[7] https://ieeexplore-ieee-org.focus.lib.kth.se/stamp/stamp.jsp?tp=&arnumber=4052584, E. Arisholm, H. Gallis, T. Dyba, D. Sjoberg. Publicerad: 2007.
[8] https://www-sciencedirect-com.focus.lib.kth.se/science/article/pii/S0164121205000038, M. Müller. 2005.

undersökningens resultat samt journalen från IEEE som nämndes innan och dessutom den sistnämnde från ScienceDirect.

5.2 Synkning

Resultaten från delkapitel 4.2 *Synkning* visar på att i fallen där deltagarna i det valda paret bestod av personer med en mer teknisk bakgrun krävde en genomsnittlig tid på sju minuter. Medans blandade par och par med mindre teknisk bakgrund krävde ytterligare tid, med ett snitt på 18 och 12 minuter.

Som beskrivet i kapitel 2.6 *Våra projekt* utfördes 2 olika webbaserade projekt. Dessa projekt krävde liknande förkunskaper vilket skapade stora likheter angående hur mycket personer med skiljande teknisk bakgrund i det valda ämnet besatt. Detta skapade stora fördelar för de personer som tidigare arbetat med liknande tekniker eftersom det krävde mindre tid för att utföra synkning inom paret. Personer som tidigare inte arbetat med liknande programmering behövde övervinna en betydande inlärningskurva innan de kunde arbeta på en likartad nivå. Detta kan bevittnas i hur tiden för synkning skiljer sig mellan de olika paren. Från denna skilljelse är det även möjligt att undersöka hur mycket den genomsnittliga tiden förändrades mellan de olika paren. Resultaten påvisar att de blandade paren samt paren med lägre kunskap inom ämnet minskade sin genomsnittliga tid under projektets gång. Detta kan bero på att under tiden som projektet utförs erhåller deltagarna mer kunskap vilket gör det enklare för dem att utföra arbetet.

Synkning är något som beror på ytterligare saker utanför tekniska kunskaper. Olika faktorer som mänskligt välmående och kommunikationskunskaper kan spela stor roll i hur väl man kan utföra synkning mellan två individer. Detta är något som kan påverka resultaten men även är svårt att undvika. Då detta kriterium endast fokuserade på faktorn angående personens bakgrund inom tekniska kunskaper är detta något man kan undersöka ytterligare i kommande rapporter.

Forskning utförd av Göteborgs universitet visar på att par som utför parprogrammering kräver en betydligt lägre tid för att utföra arbetet när paren har en minimal skillnad i tekniska kunskaper. När tekniska kunskaper skiljer sig på en högre nivå börjar par programmeringen att brytas ner enligt samma forskning. Detta liknar resultaten som rapporterats i denna rapport där paren med skiljande nivå i kunskaper krävde betydligt mer tid för att utföra arbetet jämfört med par vars kunskaper, vare sig höga eller låga, inte skiljde sig betydligt.[9]

[9] https://gupea.ub.gu.se/bitstream/2077/38591/1/gupea_2077_38591_1.pdf, Xinran He, Yuwei Chen, publicerad vid Göteborgs Universitet 2014

5.3 Förare

För detta kriterium gällde det att bedöma hur personen som agerade förare upplevde utförandet av parprogrammering utifrån förarens perspektiv. Syftet med detta kriterium var att evaluera beteendet av hur en programmerare utför ett arbete i en roll som sedan skulle evalueras av en annan person. Med andra ord skulle det granskas hur parprogrammering påverkar föraren rent allmänt samt att denne person har en som övervakar en.

I denna undersökning har förarna som varit närvarande under en session av parprogrammering fått besvara med en siffra mellan noll och tio om hur denne person har känt att det har gått efter varje sittning. Där noll är väldigt dåligt och tio är väldigt bra. Det sammanställdes en tabell med resultaten som kan observeras under rubriken *4.3 Förare* (figur 4.3). I denna kan man kontrollera att det lägsta talet en förare angav var sju. I efterhand har ett medelvärde beräknats. Det har skett tio sittningar av parprogrammering totalt. Med värdena från tabellen blir medelvärdet 8,6. Detta är ett positivt resultat eftersom detta tyder på att förarna har uppskattad parprogrammering med tanke på det högre medelvärdet.

I en rapport skriven av Danielle L. Jones och Scott D. Fleming, publicerat i IEEE 2013, kan man läsa att föraren tenderar att jobba mer avslappnat.[10] Detta eftersom författarna nämner ett experiment där förarna har berättat samt att forskarna insett att navigatörerna bidrar med mycket hjälp som underlättar arbetet för förarna. Förarna har uppskattat navigatörernas bidrag till framgång för uppgiften.

I denna undersökning har det intervjuats ett antal förare som kommit med vissa konkreta synpunkter gällande hur det känns att arbeta i rollen som förare i parprogrammering. Vissa har uppgett att det blir problematiskt när navigatören inte har liknande programmeringsvana eller förståelse för uppgiften. Detta leder till att arbetet inte sker i snabb takt. Andra förare har uppgett att de tycker att det är bra när de får förklara vissa saker som navigatören inte förstår då det bidrar till att båda parterna förstår konceptet bättre.

[10] https://ieeexplore-ieee-org.focus.lib.kth.se/stamp/stamp.jsp?tp=&arnumber=6645252, Danielle L. Jones och Scott D. Fleming, publicerat i IEEE 2013.

5.4 Observatör/Navigatör

Resultaten från avsnitt 4.4 *Observatör/navigatör* visar på att observatören anses betygsätta sig 7.2 på en skala från ett till tio angående hur de kände att arbetet gick samt hur mycket den kände att det bidrog till arbetet.

Om man analyserar den inledande delen av tabell 4.4 kan man observera att genomsnittsvärdet är betydligt mycket lägre än de kommande veckorna. En anledning till detta kan vara på grund av hur detta är ett nytt system för deltagarna att arbeta. Observatörens roll i arbetet har en mer teoretisk del eftersom att det krävs att den noggrant granskar skriven kod vilket kräver både god förståelse av koden samt tydlig förståelse av vad målet med arbetet är. Då detta är ett nytt tankesätt för deltagarna att arbeta på kan det skapa problem med hur aktivt man upplever att man deltar i arbetet under projektets inledande fas. Nästkommande del av tabellen visar på en betydligt större värdering av delaktighet i arbetet utifrån observatörens perspektiv. Detta kan, likt tidigare analys, vara på grund av att observatören under denna tidpunkt införskaffat sig tillräckligt med erfarenhet av att arbeta inom rollen vilket skapar mer förtroende om en själv.

Likt analysen i avsnitt 5.2 kan tekniska kunskaper spela en stor roll i hur resultaten uppfattas. Eftersom att det krävs att observatören finner fel eller förbättringar i förarens kod måste denne besitta likgiltiga eller mer tekniska kunskaper som föraren för att utföra ett grundligt arbete som observatör. Denna faktor kan ha haft en verkande roll i vad som visat sig i resultaten. Något som kan skapat ett säkrare resultat var om man använde indelningarna av par som skapats inför kriteriet angående *synkning* bestämde det genomsnittliga värdet mellan de olika paren istället för de totala paren.

I rapporten "*Pair programming*" skriven av Laurie Williams(North Carolina State University) undersöks par programmeringens effekt inom utbildningen, ett scenario likt det som denna rapport undersöker. Rapporten kommer med resultat som visar att parprogrammering skapar en miljö med utökad interaktion mellan studenter inom arbetet. Detta tyder på att både observatören aktivt kommunicerar sina tankar med föraren vilket kan ses i resultaten från denna rapport i hur observatören uppfattat sig själv som mer aktiv under projektets gång. Olika individuella önskemål kring olika arbetssätt bör dock inkluderas då det existerar medlemmar som föredrar att arbeta individuellt gentemot att arbeta i par.[11]

[11] https://collaboration.csc.ncsu.edu/laurie/Papers/ESE%20WilliamsPairProgramming_V2.pdf , Laurie Williams, publicerad vid CSU NCSU 2003

5.5 Effektivitet/lönsamhet

Detta kriterium utvärderades genom att jämföra den uppskattade tiden gentemot den verkliga tiden som krävdes för att utföra olika arbeten/uppgifter med hjälp av parprogrammering. Syftet med kriteriet var att i något mån uppnå ett ungefärligt värde på hur effektivt parprogrammering är när ställd emot normal individuell programmering.

Utifrån diagrammet i avsnitt 4.5 *Effektivitet/lönsamhet* kan det analyseras att den uppskattade tiden att utföra uppgifter med hjälp av parprogrammering alltid översteg den verkliga den som faktiskt krävdes. Den totala genomsnittliga felmarginalen ger ett värde på 13.7% för den uppskattade tiden. Detta kan tyda på hur effektivt parprogrammering egentligen är i och med att den uppskattade tiden skiljt sig med nästan 15%. Det finns dock andra faktorer som kan påverka detta resultat, som till exempel tidigare erfarenhet samt storleken på arbetet. Om deltagaren som agerar som förare har goda tekniska kunskaper kan den verkliga tiden för arbetet minska i stor omfattning. Detta kommer i sin tur inte tyda på att parprogrammering deltog i hur snabbt arbetet utfördes.

Som nämns ovan finns det ett flertal faktorer som spelar roll i hur effektivt parprogrammering verkligen blir. Utvärderingskriterierna som undersöks i denna rapport kan även inkluderas som faktorer som bidrar till effektiviteten av parprogrammering. Synkning för paret har inkluderats som en del av det arbetet som tar tid och även korrekthet tar tid av arbetet ju lägre korrekt koden blir. Generellt har projektet gynnat från parprogrammering enligt åsikter samlade från deltagarna men ytterligare undersökning kan krävas. Om projektet innefattade en större mängd deltagare hade det existerat en möjlighet att jämföra tiden att utföra liknande uppgifter med individuell programmering och ställa den mot tiden det tar för parprogrammering.

I rapporten “Pair programming what’s in it for me” skriven av Andrew Begel och Nachiappan Nagappan utfördes en undersökning som samlade in data från 487 deltagare som bestod av olika grupper från programmerings branschen så som testare och debuggers..
Undersökningen innehöll en del som bad deltagarna att lista deras tre största fördelar med parprogrammering. De tre mest frekventa svaren var färre buggar, delar kod kunskap samt bättre kvalite på koden. Alla dessa faktorer bidrar till mer effektivt programmeringsarbete vilket kan dra likheter med undersökningen som denna rapport även utförde. Detta ger dock inget direkt värde på effektiviteten av parprogrammering samt det tar inte hänsyn till olika faktorer som kan ha en negativ inverkan på arbete med parprogrammering. Sådana faktorer kan innefatta schemaläggning, personlighetsproblem samt kunskapsskillnader.[12]

[12] https://www.researchgate.net/publication/221494979_Pair_Programming_What's_in_it_for_Me ,Andrew Begel et al, publicerad vid ESEM 2008

6. Slutsats och framtida arbete

Det finns flera olika arbetsmetoder samt verktyg att implementera i diverse projekt inom mjukvaruutveckling. Under denna undersökningens genomförande har det framkommit mycket relevant och gynnsam information samt viktiga upplysningar kring ämnet parprogrammering. Det har konstaterats ett antal olika fördelar och nackdelar med denna metod av arbete.

Målet med denna forskning var att utvärdera och påvisa hur effektivt parprogrammering egentligen är. Undersökningen baserades på arbeten inom mjukvarurelaterade projekt där denna metod var tillämpad.

Generellt har en positiv uppfattning kring parprogrammering inträffat. Det har framkommit fördelar som att förarna känner välbehag då en navigatör underlättar arbetet. En annan fördel var att parprogrammering resulterade i en större grad av korrekthet. En nackdel som framkom var att det i vissa fall gick långsamt då förare och navigatör inte var på samma nivå av erfarenhet eller olika uppfattningar kring ett ämne.

Inför framtida arbeten skulle det vara bra om liknande undersökningar genomfördes dock i andra miljöer. Till exempel ska dessa tester genomföras i en grupp där medlemmarna har hög grad av erfarenhet när det gäller programmering samt att arbeta i projekt. Detta skulle bidra till en mer utvidgad syn kring ämnet.

7. Referenser

A. Aitken and V. Ilango, "A Comparative Analysis of Traditional Software Engineering and Agile Software Development," *2013 46th Hawaii International Conference on System Sciences*, 2013, https://ieeexplore.ieee.org/abstract/document/6480417, (hämtad 2020-05-06).

Andrew Begel, "Pair Programming: What's in it for Me?", ESEM, 2008 https://www.researchgate.net/publication/221494979_Pair_Programming_What's_in_it_for_Me, (hämtad 2021-05-20).

D. Marks, "Development Methodologies Compared", N CYCLES software solutions, December 2002 , www.ncycles.com, (hämtad 2020-05-10).

Danielle L. Jones och Scott D. Fleming, What Use Is a Backseat Driver? A Qualitative Investigation of Pair Programming, IEEE, 2013, https://ieeexplore-ieee-org.focus.lib.kth.se/stamp/stamp.jsp?tp=&arnumber=6645252, (hämtad 2020-05-19).

E. Arisholm, H. Gallis, T. Dyba, D. Sjoberg, "Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise", IEEE, 2007 https://ieeexplore-ieee-org.focus.lib.kth.se/stamp/stamp.jsp?tp=&arnumber=4052584, (hämtad 2020-05-17).

L. Williams, "Pair Programming", CSU NCSU, 2003, https://collaboration.csc.ncsu.edu/laurie/Papers/ESE%20WilliamsPairProgramming_V2.pdf (hämtad 2021-05-10)

M. A. Awad, A Comparison between Agile and Traditional Software Development Methodologies, Academia, 2005 https://www.academia.edu/download/58993716/10.1.1.464.609020190422-13963-j0ju8a.pdf, (hämtad 2020-05-12).

M. Müller, "Two controlled experiments concerning the comparison of pair programming to peer review", ScienceDirect, 2005, https://www-sciencedirect-com.focus.lib.kth.se/science/article/pii/S0164121205000038, (hämtad 2020-05-17)

W. Sun, G. Marakas, M. Aguirre-Urreta,The Effectiveness of Pair Programming: Software Professionals' Perceptions, IEEE, 2016, https://ieeexplore-ieee-org.focus.lib.kth.se/document/7274252, (hämtad 2020-05-11).

Yvonne Dittrich et al, Exploring the evolution of software practices, Association for Computing Machinery, 2020, https://dl.acm.org/doi/pdf/10.1145/3368089.3409766, (hämtad 2020-05-07).

Xinran He, Yuwei Chen, "Analyzing the Efficiency of Pair Programming in Education", University of Gothenburg, 2014, https://gupea.ub.gu.se/bitstream/2077/38591/1/gupea_2077_38591_1.pdf, (hämtad 2020-05-18).