TEKNISKA HÖGSKOLAN

HÖGSKOLAN I JÖNKÖPING

Testdriven utveckling – En utvärdering av Jordbruksverkets metod för systemutveckling

Test Driven Development- An evaluation of the Board of Agriculture's method for system development

Marie Björk

EXAMENSARBETE 2012

Datateknik

Postadress:
Box 1026
551 11 Jönköping

Besöksadress:
Gjuterigatan 5

Telefon:
036-10 10 00 (vx)

Detta examensarbete är utfört vid Tekniska Högskolan i Jönköping inom ämnesområdet Datateknik. Arbetet är ett led i den treåriga högskoleingenjörsutbildningen
Författarna svarar själva för framförda åsikter, slutsatser och resultat.

Examinator: Inger Palmgren

Handledare: Inger Palmgren

Omfattning: 15 hp

Datum: Juni 7, 2012

Postadress:	Besöksadress:	Telefon:
Box 1026	Gjuterigatan 5	036-10 10 00 (vx)
551 11 Jönköping		

Abstract

In spring 2010 was a comprehensive report made regarding the Board of Agriculture's test and quality work in their development process. The report identified several strengths and improvements on the test process. In order to address several of the weaknesses was decided that a test-driven development approach could be used for new development of the Board of Agriculture's new client system Kund. Test Driven Development is a relatively new development method that belongs to the family of agile methods. TDD is said to lead to, among other things, increased code quality and reduced development time by so-called unit tests are written and performed before the actual program code is written. Board of Agriculture has no previous experience at this stage of the development of TDD, and that the method has not yet been evaluated based on the Board's conditions, so it is uncertain whether the benefits described in theory may occur when applying TDD.

The purpose of this study was to examine the suitability of TDD as a development method for system development project and future system management for the Board of Agriculture. The goal have been to give an idea of the method's pros and cons and what the expectations are, and to capture relevant information and experiences that can be carried forward and help at the next project and future management.

Three part questions regarding: (1) Named theoretical characteristics based on literature studies, (2) the Board's expectations of the method, and (3) How well reaches the method up to the stated expectations, were set up to answer the primary question: *How suitable is the test-driven approach the Board of Agriculture chosen to apply for the pilot project "Kund", for systems development?*

To answer these questions, the theory primarily came from literature reviews, independent empirical studies, and documentation has been provided by the Board of Agriculture. To confirm the practical proofs an interview were conducted on the developers and test managers in the project.

The results from the data collection showed several features that the Board expects from the method. These were primarily in the field of quality growth and reduced development time, but it also showed that tests needs be performed earlier in the process and faults needed to be found at an earlier stage. The interview study was designed after these results and the participants responded freely around several assertions. When analyzing the collected material showed results that TDD over time can generate many of the improvements described in the theory when applying. Another conclusion that could be drawn was that TDD has the potential to meet the expectations of productivity and code quality that the Board of Agriculture has set for the method, but this depends strongly on if the developers has time to learn and to gain experience of the method.

Keywords: *System Development, Agile systems development methods, Test Driven Development, The Board of Agriculture*

Sammanfattning

Våren 2010 lades en omfattande rapport fram angående Jordbruksverkets test- och kvalitetsarbete inom systemutvecklingsprocessen. Där identifierades flera styrkor och förbättringsåtgärder kring testarbetet. För att åtgärda flera av svagheterna togs beslutet att en testdriven utvecklingsmetod skulle användas vid nyutvecklingen av Jordbruksverkets nya kundsystem Kund. Testdriven utveckling är en relativ ny utvecklingsmetod som hör till familjen agila metoder, som betyder lättrörlig. TDD sägs leda till bland annat ökad kodkvalité och minskad utvecklingstid genom att så kallade enhetstester skrivs och utförs innan själva programkoden skrivs. Jordbruksverket saknar i nuläget tidigare erfarenhet av att utveckla enligt TDD, samt att metoden inte ännu har utvärderats utifrån Jordbruksverkets förhållanden vilket gör att det är oklart om de fördelar som beskrivs i teorin kan komma att inträffa vid en införing av TDD.

Syftet med examensarbetet har varit att undersöka TDD:s lämplighet som utvecklingsmetod inom projekt och förvaltning på Jordbruksverket. Målet har varit att ge en bild av metodens för-och nackdelar samt vilka förväntningar som finns, samt att fånga upp relevant information och erfarenheter som kan föras vidare och hjälpa vid nästkommande projekt och förvaltning.

Tre delfrågeställningar angående: (1) angivna teoretiska egenskaper utifrån litteraturstudier, (2) Jordbruksverkets förväntningar på metoden och (3) hur väl når metoden upp till angivna förväntningar, sattes upp för att kunna svara på den primära frågeställningen: *Hur lämplig är den testdrivna metod som har valts att tillämpas för pilotprojektet "Kund", för systemutveckling inom Jordbruksverket?*

För att få svar på dessa frågeställningar har teorin främst kommit ifrån litteraturstudier, oberoende empiriska undersökningar, samt dokumentation som har tillhandahållits av Jordbruksverket. För att bekräfta de teoretiska egenskaperna utfördes också en intervjustudie på utvecklare och testansvariga inom projektet.

Resultaten från datainsamlingen visade flera egenskaper som Jordbruksverket förväntar sig av metoden. Dessa var främst inom området kvalitétsökning och minskad utvecklingstid, men även att test behöver komma tidigare in i processen och att buggar skall komma att åtgärdas i ett tidigare stadie. Intervjustudien utformades efter detta resultat och deltagarna fick svara fritt kring flera påståenden.

Vid analys av insamlat material framkom resultatet att TDD över tid kan generera många av de förbättringar som beskrivits i teorin vid tillämpning. En annan slutsats som kunde dras var att TDD har potential att uppfylla de förväntningar inom produktivitet och kodkvalitet som Jordbruksverket har satt upp för metoden, men att detta är starkt beroende av att utvecklarna får tid på sig att lära sig och skaffa sig erfarenhet av metoden.

Nyckelord

Systemutveckling, Agila systemutvecklingsmetoder, Testdriven utveckling, Statens jordbruksverk

Innehållsförteckning

1 Inledning

Statens jordbruksverk är Sveriges expertmyndighet för landets jordbruks- och livsmedelspolitik. Dess mål är att:

”Verka för en konkurrenskraftig, miljö- och djurskyddsanpassad livsmedelsproduktion till nytta för konsumenterna och att skapa förutsättningar för ett livskraftigt jordbruk i mindre gynnade områden.” [1]

Jordbruksverket har också rollen som den centrala och samordnade myndigheten för EU:s gemensamma jordbrukspolitik. Jordbruksverket administrerar och tillämpar därför EU:s regleringar och stödsystem på jordbruksområdet. [2]

Eftersom Jordbruksverket är en utbetalande myndighet sätter detta stora krav på korrekta IT-lösningar. Det största kravet ligger på systemens kvalitet, rätt uppgifter måste visas vid rätt tillfälle.

Därför måste Jordbruksverket hela tiden anpassa sig för att kunna ligga i framkant bland de statliga myndigheterna när det gäller IT-lösningar, och dessa måste bland annat uppnå tillgänglighet, stabilitet och vara framtidssäkra. [3]

1.1 Bakgrund och problembeskrivning

Våren 2010 gjordes en omfattande undersökning kring Jordbruksverkets test- och kvalitetsarbete inom systemutvecklingsprocessen, även kallad Health check. Syftet med rapporten var att identifiera styrkor och förbättringspotential kring testarbetet, samt att öka medvetenheten vid test. Rapporten presenterade då flera förbättringsåtgärder som Jordbruksverket har tagit till sig och arbetar med dessa i nuläget.

För att kunna lösa flera av åtgärderna gällande testarbetet valdes den testbaserade agila metoden Testdriven utveckling(TDD) vid nyutvecklingen av Jordbruksverkets nya kundsystem Kund. Utvecklingen av systemet är ett pilotprojekt till den nya utvecklingsplattformen som är under utveckling.

TDD är ett nytt arbetssätt för samtliga utvecklare och tidigare erfarenhet från att programmera enligt metoden saknas. Metoden har ännu inte utvärderats utifrån Jordbruksverkets förhållanden vilket gör att det är oklart om de fördelar som beskrivs i teorin kan komma att inträffa vid en tillämpning av TDD på jordbruksverket.

1.2 Uppdrag och mål

Uppdraget är att genom att kombinera tillgänglig teori med empiri från Jordbruksverket, utvärdera lämpligheten i TDD som utvecklingsmetod inom projekt och förvaltning.

Målet är att kunna presentera en väl grundad rapport på metodens för- och nackdelar samt vilka förväntningar som finns, för att kunna ge en slutsats i TDDs lämplighet som utvecklingsmetod på Jordbruksverket.

1.3 Syfte och frågeställningar

Syftet med examensarbetet är att undersöka om TDD är en lämplig utvecklingsmetod på IT-avdelningen, med fokus på utvecklingsfasen. Detta är för att fånga upp relevant information och erfarenheter som kan föras vidare och vara till hjälp vid nästkommande projekt och förvaltning under dess utvecklingsarbete.

Följande frågeställning kommer att ligga grund för detta examensarbete:

Hur lämplig är den testdrivna metod som har valts att tillämpas för pilotprojektet "Kund", för systemutveckling inom Jordbruksverket?

För att svara på ovanstående fråga inom detta examensarbete har tre delfrågeställningar satts upp:

1. Vilka är de mest framträdande egenskaper som kan identifieras i samband med användande av TDD i utvecklingsprocessen?
2. Vilka egenskaper/förbättringar förväntar sig Jordbruksverket från TDD?
3. Finns det stöd för att TDD klarar av att motsvara de krav som Jordbruksverket har satt upp, samt går det att påvisa att metoden kan leverera de praktiska fördelar som den påstås medföra?

1.4 Avgränsningar

Examensarbetet kommer bara att behandla själva utvecklingsarbetet inom projektet och de moment som ingår. Övrig projektledning, styrning, kravfångst, implementering och drift kommer att lämnas utanför för att minska omfånget av uppsatsen.

1.5 Disposition

Rapporten följer den mall som Högskolan i Jönköping tillhandahåller för examensarbeten.

I teoretisk bakgrund kommer läsaren att introduceras till den teori som kommer att ligga till grund för uppsatsen.

I metod och genomförande beskrivs det tillvägagångssätt som har använts för att kunna svara på ovan nämna frågeställningar.

I resultat och analys presenteras de resultat som har framkommit och hur dessa stämmer överens med tidigare framtagna mål.

I diskussion och slutsatser diskuteras det analyserade resultatet och utvärderas utifrån syfte och frågeställningar.

2 Teoretisk bakgrund

Detta avsnitt kommer att fördjupa läsaren i de teorier och koncept som är förankrade till frågeställningarna.

2.1 Terminologi

Enhetstest - Enhetstester är de tester programmeraren själv skriver för att verifiera den egna koden. En enhet definieras här med att det är den minsta delen av ett system som är testbar. [4]

Automatiserad enhetstest – Enhetstester som körs automatiskt vid varje kompilering. [4]

User Storie- Även kallad storie, en deluppgift eller funktionalitet som skall utvecklas.

Mockning – Ett dataobjekt som isoleras från delar som enheten är beroende av. Detta för att kunna testa varje enhet var för sig. [4]

Exempel: En koppling till skatteverket för hämtning av personuppgifter kan tas bort tillfälligt för att kunna utföra tester i det egenutvecklade systemet.

Artefakter – Är den källkod i form av en komponent som är incheckad och klar för att byggas ihop och testas tillsammans med övriga artefakter. [4]

2.2 Agil systemutveckling

Ordet agil är en svensk översättning från Agile som betyder lättrörlig. Grundtanken med agil systemutveckling är att omvärlden alltid förändras och utvecklingsmetoder måste kunna följa med i dessa förändringar och anpassas [5] [6].

För att skapa mer förstående för agil systemutveckling bildade en grupp av experter *Agile Alliance*. Dessa experter definierade upp 12 principer för agil systemutveckling som heter *Agile Manifesto.* [7, 8]

1. Högsta prioritet är att tillfredsställa kunden genom tidiga och kontinuerliga leveranser av värdefull programvara.
2. Välkomna förändrade krav, även sent u utvecklingen. Agila metoder utnyttjar förändring till kundens konkurrensfördel.
3. Leverera funktionell mjukvara frekvent, från ett par veckor till ett par månaders mellanrum.
4. Verksamhetskunniga och systemutvecklare måste jobba tillsammans dagligen genom projektets gång.
5. Bygg projekt kring motiverade personer. Ge dem den miljö och stöd dem behöver, och lita på att de får jobbet gjort
6. Den metod som är mest tillräcklig och mest effektiv för kommunikation inom och utanför utvecklingsteamet är ansikte-mot-ansikte - konversation.
7. Fungerande programvara är det primära måttet på framsteg.
8. Agila metoder verkar för hållbar utveckling. Sponsorer, utvecklare och användare skall kunna hålla en konstant takt på obegränsad tid.
9. Kontinuerlig uppmärksamhet ny teknik och design stärker anpassningsförmågan.
10. Enkelhet - Konsten att maximera mängden arbete som inte genomförs är grundläggande.
11. Den bästa arkitekturen, krav och design kommer från själv organiserade team.
12. Att vid jämna mellanrum, reflektera över hur mer effektiv teamet kan bli, och sedan anpassa beteendet därefter.

2.3 Testdriven utveckling (TDD)

Testdriven utveckling är en relativt ny metod som tidigare bara har använts sporadiskt genom åren, men har som medlem bland de agila utvecklingsmetoderna fått ökad uppmärksamhet. Även om TDD anses som en betydande del i metoden Extreme programming(XP), har metoden fått mycket enskild uppmärksamhet. Både utvecklingsverktyg och böcker har för specifikt TDD publicerats samt öppnat dörrar för undersökningar och utvärderingar [9].

Metoden kombinerar två generella principer:

1. Skriv ner testfall innan kodning.

2. Gör dem genomförbara för tester [5, 10].

Kent Beck som är grundaren till XP har definierat TDD med två enkla regler [7, 11].

Skriva bara ny kod om en testkod har fallerat.

Eliminera alla dubbletter i den skrivna koden.

2.3.1 TDD som process

Inom TDD skapar programmeraren både testerna och programkoden genom korta iterationer [9, 12].

1. Några få enhetstester skapas för en mindre uppgift av en storie, dessa kommer att fallera på grund av att det inte ännu finns någon programkod.
2. Programkod som implementerar uppgiften skrivs. Detta är för att kunna passera enhetstesten
3. Samtliga tester körs för att ge en försäkran om att enhetstesterna kan passera tillsammans med ny kod.
4. För att förhindra prestandaproblem och dubbletter struktureras programkoden om, så kallad refraktionering.
5. Efter refraktionering måste alla tester köras igen för att koden skall kunna valideras.

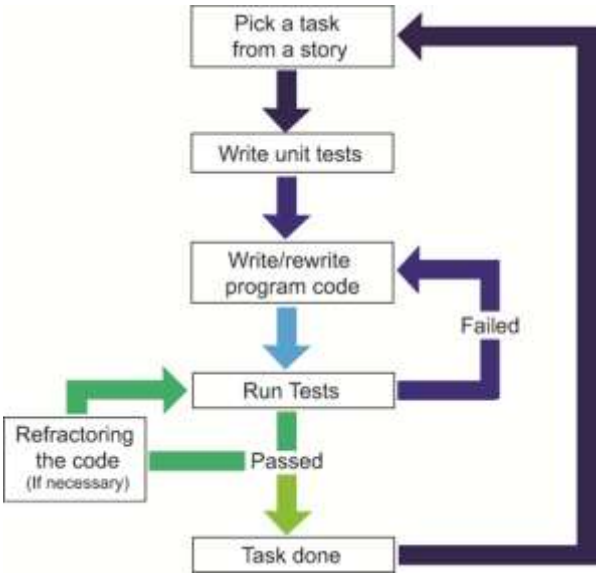

Fig. 1 Överblick av TDD-processen enl. Erdogmus H, Morisio M.[12]

2.3.2 Fördelar med TDD

TDD anses föra med sig flera fördelar i utvecklingsarbetet. Det finns flera undersökningar som tyder på detta, men än så länge har det inte framkommit något entydigt resultat. Följande nyttor är definierade av Erdogmus H och Morisio M. [12]

Feedback: Testen skall ge programmeraren konstant feedback om huruvida ny funktionalitet har blivit implementerad som tänkt och om den påverkat tidigare funktionalitet.

Uppgiftsorienterad: Testen driver kodningen, och ger programmeraren möjligheten att bryta ner problem till mindre, mer hanterbara uppgifter vilket hjälper att behålla fokus.

Kvalitetssäkring: Eftersom alla enhetstesterna måste klara testerna, innan ny kod implementeras ger det en verifikation om att den nya koden inte har genererat några fel. Detta säkrar en viss kvalitet som sedan kan underhållas av testerna.

Låg nivå på design: Tester kan hjälpa till i designbeslut på låg nivå, genom att bestämma vilka klasser som skall skapas och dess namnsättning.

Vidare har även B George och Laurie W definierat nyttor de tror TDD för med sig. [13]

Programförståelse: TDD hjälper utvecklaren att förklara sin kod genom tester istället för att beskriva genom ord.

Effektivitet: Eftersom metoden ger utvecklaren kontinuerlig feedback i sin programmering, samt att defekter hittas tidigare gör det att utvecklaren kan arbeta med effektivt.

Tester: Genom de automatiserade enhetstesterna ger det bättre kvalitet, förbättrar testarbetet och reducerar arbetstid för manuella tester.

Minskat arbete för små åtgärder: En mindre åtgärd som borde ha uppkommit kan komma att kosta mer tid och pengar än förväntat.

2.3.3 Nackdelar med TDD

Enligt B. George och L. Williams fanns det också några nackdelar med metoden som borde tas hänsyn till [14]. Dessa var:

Praktisk tillämpning: Tillämpning av TDD kan kräva mycket arbete för att kunna "mocka bort" objekt, för att kunna göra dem testbara.

Tillit på refraktionering: Inom TDD finns det en stor tillit till refraktionering, som skall hjälpa till att underhålla systemet och för att undvika komplexitet. Därför måste refraktionering fungera till fullo.

Kompetens: För att kunna skriva testfall för svårtestad kod kan det behöva en större kompetens än vad en oerfaren utvecklare har, vilket kan resultera i att funktionaliteten inte blir korrekt testad.

2.3.4 Kodexempel

När en bit av funktionalitet skall implementeras skrivs första koden som skall testa funktionaliteten. På detta sätt bestämmer testen vad för kod som behöver skrivas. Utvecklaren skriver bara det som behövs för att koden skall passera testet, inte mer. Nedan är ett kodexempel i Java-miljö av en lista med filmer och dess betygsättning. [6]

Först skrivs testet genom att göra ett påstående som skall vara sant:

```
public void testRating(){
      assertEquals("Bad average rating.",
4,starWars.getAverageRating());
}
```

Detta är inte tillräckligt, utan det behövs också ett underlag för att påståendet skall kunna bli sant, och då behövs det ett betyg till filmen.

```
Public void testRating(){
      starWars.addRating(3);
      starWars.addRating(5);
      assertEquals("Bad average rating.",4,starWars.getAverageRating());
}
```

Slutligen måste förekomsten av Movie deklareras.

```
Public void testRating(){
      Movie starWars = new Movie("Star Wars");
      starWars.addRating(3);
      starWars.addRating(5);
      assertEquals("Bad average rating.",4,starWars.getAverageRating());
}
```

Vid kompileringen reagerar testet på att addRating(int) och getAverageRating() är odefinierade. Följande kod behöver läggas till i Movie.

```
public void addRating(int newRating){
}
public int getAverageRating(){
      return 0;
}
```

Vid utveckling i Java måste ett returvärde finnas eftersom getAverage returnerar en integrer. Nu kompileras koden med testet går inte genom. Tidigare angavs

genomsnittsvariabeln, getAverage till 4, men i senaste exemplet angavs den till 0. Anges getAverageRating till 4 kommer testet att gå genom.

```
public void addRating(int newRating){
}
public int getAverageRating(){
      return 4;
}
```

Som nästa steg skall koden refraktionera för att ta bort dubbletter och onödig kod. I exemplet har filmen fått två olika betyg 3 och 5. Genomsnittet har tidigare angetts som 4. Istället för ett fastvärde kan en variabel användas för att ange värdet. 4:an är då en dubblett som kan ersättas.

```
public int getAverageRating(){
      return (3+5)/2;
}
```

Koden passerar testet, men koden kan fortfarande förbättras. Bland annat genom att ta bort 3 och 5 genom att lägga till en variabel för dem.

```
private int totalRating = 0;
}
public void addRating(int newRating){
      totalRating += newRating;
}
public int getAverageRating(){
      returntotalRating/2;
}
```

Som slutgiltigt steg i refraktionering tas även täljaren 2 bort och ersätts med en variabel.

```
private int totalRating = 0;
private int numberOfRatings = 0;

public void addRating(int newRating){
      totalRating += newRating;
      numberOfRatings ++;
}
public int getAverageRating(){
      return totalRating/numberOfRatings;
}
```

Koden kan inte byggas om mer och kan kompileras en sista gång. När koden har klarat testet är koden klart.

2.3.5 Tidigare empiriska undersökningar

Även om TDD har på senare tid börjat användas mer flitigt, så finns det få empiriska undersökningar som kan styrka de fördelar som metoden påstås ge. [8, 11]

2.3.5.1 Hagner och Müller [10]

Hagner och Müllers experiment bestod av 19 studenter inom datateknik vid universitetet i Karlsruhe, Tyskland. Dessa studenter hade studerat i tre år samt genomgått en kurs i utvecklingstekniker kring XP. Studenterna delades in i två grupper. En grupp använde sig av en testdriven metod, medans den andra gruppen använde den traditionella metoden, där koden implementeras först för att sedan genomföra tester.

Fokus under experimentet låg på effektivitet, kvalitet och möjlighet till återanvändning av kod. Syftet var att hitta för och nackdelar som kan uppstå för en utvecklare om denne skulle byta till TDD.

Uppgiften bestod av att implementera en huvudklass i en applikation som saknade body, men hade alla deklarationerna klara. Studenterna skulle utveckla och utföra tester tills det att de ansåg sig vara klara med uppgiften. Studenterna använde sig av Java med jUnit under utvecklingen.

Genom experimentet kom Hagner och Müller fram till följande resultat:

(1) Om en utvecklare byter från traditionell utveckling till testdriven, så behöver inte utvecklingen nödvändigtvis gå fortare.

(2) TDD lönar sig bara en aning gällande ökad pålitlighet. Resultatet var väldigt otydligt och en klar slutsats kunde inte dras.

(3) Gruppen som använde TDD kunde återanvända redan existerande programmetoder både fortare och mer effektivt. Detta tros bero på att

programförståelsen ökade, och utvecklaren kunde också lätt lära sig hur metoderna används.

2.3.5.2 Maximilien, Williams och Vouk [14]

Denna studie undersökte ett utvecklingsteam på IBM. Studien följde utvecklingsarbetet kring en av IBMs större program där TDD skulle tillämpas inför en ny version av programmet. Alla inom projektteamet var erfarna programmerare, men utan någon direkt erfarenhet av TDD.

Studien hade fokus på effektiviteten och dess relation till defektreducering jämfört med traditionella metoder. För att kunna undersöka antalet defekter vad den implementerade koden tvungen att passera acceptanstester som utfördes av en extern testgrupp.

Resultatet visade att genom att använda TDD reducerades 40 % av defekterna jämfört med traditionella metoder. Projektet blev också slutfört i tid, men utan någon större inverkan på produktiviteten. Ytterligare kunde de automatiserade enhetstesterna bli en tillgång som kunde fortsätta att förbättras i framtiden.

2.3.5.3 George och Williams [13]

George and Williams experiment bestod av grupper om åtta personer vid företagen John Deere, RoleModel Software och Ericsson. Alla var erfarna programmerare men hade varierande erfarenhet av TDD som utvecklingsmetod. Deltagarna delades mellan en kontrollgrupp som använde sig av den mer traditionella vattenfallsmetoden, och den andra gruppen använde TDD tillsammans med par-programmering. Uppgiften var att utveckla ett litet Bowlingspel i Java som de utvecklade två och två, medans kontrollgruppen utövade vattenfallsmetoden.

Experimentets hypoteser var:

(1) TDD genererar bättre kodkvalité jämfört med traditionell utveckling så som en vattenfalls-liknande modell.

(2) Programmerare som använder sig av TDD kan utveckla kod fortare än programmerare som använder den traditionella metoden.

I försöken skulle försökspersonerna hantera felen som uppstod på ett bra sätt, men fick inga acceptanstester, medan i kontrollgruppen fick de skriva automatiserade tester efter det att utvecklingen var gjord. Slutligen gick den implementerade koden genom flera så kallade black box-tester som försöksgrupperna inte hade vetskap om för att kunna verifiera kvalité och effektivitet.

George och Williams slutsats efter experimentet var att TDD tyder på att kunna ge bättre kvalité jämfört med traditionell utvecklingsmetod. Dock tog dess utvecklare längre tid på sig att utveckla (16 % längre tid). Efter en enkätundersökning som gjordes efter deltagandet visade på att 80 % av

programmerarna tyckte att TDD var en effektiv metod och 78 % tyckte att den förbättrade produktiviteten.

2.3.5.4 Erdogmus, Morisio och Torchiano [12]

Denna undersökning hade fokus på att finna styrkor och svagheter med TDD. Målet är att utvärdera kvalitet och utvecklarens produktivitet.

Hypoteserna var:

(1) TDD- utvecklare skriver mer tester per enhet och arbetsinsats.

(2) TDD- utvecklare producerar program med högre kvalitet.

(3) Att skriva mer tester förbättrar kvalitén.

(4) Att skriva mer tester ökar produktiviteten.

Försöksgruppen var tredjeårs studenter som följde en intensivkurs i Java och hur att skriva enhetstester i JUnit. Dessa var indelade i två grupper, en som använde TDD och den andra gruppen som använde en traditionell metod där testerna utförs efter utveckling.

Uppgifter bestod av att göra samma bowlingprogram som användes vid George och Williams undersökning [13], men här delades uppgiften upp i mindre stories. Varje storie hade ett specifikt black box-test som utvecklarna inte visste om. Dessa test kördes automatiskt för varje storie som var klar. Utöver acceptanstesterna utförde utvecklarna egna enhetstester.

Som resultat kom de fram till att en utvecklare skriver mer tester per enhet. Enligt Erdogmus, Morisio och Torchiano visar detta att en större mängd tester leder proportionellt till ökad produktivitet. De tror också att genom att genomföra ett test åt gången och innan implementation gör det lättare att bryta ner koden.

Gällande kodkvalité så uppnådde inte den grupp som använde TDD en bättre kodkvalité i genomsnittet, men de hade en mer enhetligt resultat jämfört med den andra försöksgruppen. De misstänker att detta kunde bero på utvecklarnas variation i skicklighet.

2.3.5.5 Janzen och Saiedien [9]

Janzen och Saiediens undersökning gick ut på att jämföra TDD med en traditionell linjär metod, och samtidigt utvärdera kvalitet, produktivitet och programmerarens uppfattningsförmåga.

Försökspersonerna var studenter som hade minst erfarenhet från två tidigare programmeringskurser. Uppgiften var att bygga en HTML-baserad applikation. Studenterna blev indelade i tre grupper. Två av grupperna använde sig av TDD men bara en grupp fick skriva tester innan utveckling. Denna grupp betecknades med ”test först” och den andra gruppen fick beteckningen ”testa sist” eftersom de fick test tilldelade efter utvecklingen. Den tredje gruppen fick beteckningen ”Inga test” då de inte hann med sina tester. Slutsatsen är främst baserad på resultatet från grupp ett och två.

Hypoteserna inom experimentet formulerades följande:

(1) Kan TDD-programmerare vara mer produktiva?

(2) Skriver TDD- programmerare mer tester?

(2b) Täcker testerna med funktionalitet?

(3) Har kod skapad genom TDD en högre kvalité?

(3b) Har TDD-kod som täcks av tester högre kvalitet änden TDD-kod som inte täcks av tester?

Utöver dessa hypoteser ägde en enkätundersökning rum där det undersöktes om utvecklarna tyckte att TDD är ett bättre tillvägagångssätt och om utvecklare som testat TDD föredrar metoden framför en traditionell metod.

Som resultat kom de fram till att den grupp som utförde testerna först hade implementerat dubbelt så mycket funktionalitet än den grupp som utförde testerna sist, men med likvärdigt antal defekter. Grupp ett som utförde sina tester först var den enda gruppen som hann bli klar med GUI för applikationen. Trots detta så använde inte gruppen mer tid än de andra grupperna som inte hann klart. Jämfört med grupp två som utförde testerna sist hade den första gruppen lagt 57 % mindre arbete per funktionalitet och 88 % mindre än grupp tre.

Gällande kodkvalité så skrev grupp ett dubbelt så många antagande per kodrad, men resultaten visade på att testerna bara täckte något mer funktionalitet än de andra grupperna. På det sättet gick det inte påvisa att TDD gav bättre kvalité.

Genom den enkätundersökning som genomfördes på försökspersonerna, gavs resultatet att de är mer positivt inställda på TDD efter experimentet. 89 % tyckte att TDD gav en enklare design, 70 % tyckte att metoden producerade kod med mindre defekter och 75 % tyckte att TDD var bästa tillvägagångsättet för uppgiften. Grupp ett visade även ett ökat självförtroende i att göra ändringar i sin egen kod.

3 Metod och genomförande

I detta avsnitt kommer en redogörning av de metoder och tillvägagångssätt som har används för att kunna svara på frågeställningarna:

Finns det stöd för att TDD klarar av att motsvara de krav som Jordbruksverket har satt upp, samt går det att påvisa att metoden kan leverera de teoretiska fördelar som identifieras?

1. Vilka är de mest framträdande egenskaper som kan identifieras i samband med användande av TDD i utvecklingsprocessen?
Detta skall besvaras genom den datainsamling som har skett. Detta inkluderar även resultatet från de empiriska undersökningar som har granskats.

2. Vilka egenskaper/förbättringar förväntar sig Jordbruksverket från TDD?
Detta skall besvaras genom att granska de interna dokument som Jordbruksverket har utgått ifrån för att kunna välja utvecklingsmetod, tillsammans med Health check

3. Finns det stöd för att TDD klarar av att motsvara de krav som Jordbruksverket har satt upp, samt går det att påvisa att metoden kan leverera de praktiska fördelar som den påstås medföra?
Denna delfråga skall besvaras genom att utföra intervjuer med berörda personer inom pilotprojektet. Tillsammans med resultatet från intervjuerna och den teoretiska delen skall en slutsats om metodens lämplighet kunna dras.

3.1 Datainsamling

3.1.1 Litteraturstudier

Datainsamlingen togs främst ifrån litteraturstudier som beskriver TDD och dess egenskaper i utvecklingsprocessen.

3.1.2 Empiriska studier

För att kunna finna de praktiska fördelar- och nackdelar som TDD tar med sig i systemutveckling har resultatet från fem olika oberoende undersökningar studerats. Dessa valdes ut för att de främst fokuserade på den påverkan som metoden gav i kvalitet och effektivitet inom utvecklingsfasen. De använde sig även av samma programmeringsspråk som tillämpas i Jordbruksverkets pilotprojekt, samt att de rörde stora och små försöksgrupper med både erfarna och oerfarna deltagare.

Resultatet från dessa undersökningar kunde sedan sammanställas som de praktiska fördelarna som skulle kunna förväntas av metoden, för att sedan kunna ligga till grund för utformningen av intervjuerna.

3.1.3 Dokumentation från Jordbruksverket

Här kommer den interna dokumentation som står till grund för de behov och förväntningar som Jordbruksverket har på metoden att beskrivas.

3.1.3.1 Health check [15]

Health check är en rapport som gjordes våren 2010 med syftet att få en uppfattning om hur test och kvalitetsarbetet ser ut på Jordbruksverket. Rapporten resulterade i flera svagheter som Jordbruksverket borde åtgärda för att optimera sitt testarbete. Detta är ett dokument som inte är öppen för allmänheten. För denna uppsats finns det dock ett godkännande av enhetschefen för Systemkonstruktion.

3.1.3.2 Metoddokumentation [4]

Detta är dokumentation gällande regler om hur metoden skall bedrivas på Jordbruksverket samt övriga förväntningar som finns på metoden. Även denna dokumentation är hämtad från interna dokument.

3.1.4 Intervjuer

Intervjuer ät till för att samla in information inom ett visst område och har ett bestämt syfte med utfrågningen. I vetenskapliga sammanhang nämns tre krav som en professionell genomförd intervju måste uppfylla. [16]

1. Metoden måste ge tillförlitliga resultat

2. Resultaten måste vara giltiga

3. Det måste vara möjligt för andra att kritiskt granska slutsatserna

Intervjuer valdes för att undersöka hur utvecklarna såg på utvecklings metoden och om TDD uppfyller de förväntningar som den valdes för. Objekten för intervjuerna var personer som har utvecklat inom Kund-projektet. Eftersom projektgruppen inte är så stor uteslöts en enkätundersökning

3.1.4.1 Intervjuform

Det finns olika former av intervjuer, de vanligaste är strukturerade och ostrukturerade intervjuer. I de strukturerade intervjuerna finns det ett antal färdiga frågor som skall besvaras. I mindre styrda intervjuer har intervjuaren däremot ett antal frågeområden som skall besvaras. [17]

De strukturerade intervjuerna har fasta svarsalternativ till alla respondenter, och ger möjlighet att få svar på frågor som hur mycket eller hur många en så kallad kvantitativ intervju. När frågorna är ostrukturerade får den intervjuade beskriva fritt sin uppfattning och tankar kring ett fenomen, en så kallas kvalitativ intervju [16].

För uppsatsens syfte valdes en kvalitativ intervju, men med en semistrukturerad intervjuform. Detta för att kunna följa en intervjuguide med specifika frågeområden, men samtidigt låta den intervjuade tala fritt om sin upplevelse kring

dessa Tillvägagångsmetoden för intervjuundersökningen grundas på Kvales sju stadier, som bygger på idén att se på hela intervjuprocessen och ge intervjuaren möjligheten att göra genomtänkta beslut. [18]

Tematisering: Tematisering består av att formulera undersökningens syfte och frågeställningarna inför undersökningen. Undersökningens varför och vad skallklargöras innan.

Planering: Planeringen av undersökningens upplägg görs med hänsyn till de sju stadierna och utifrån vilket resultat som önskas.

Intervju: Intervjuerna genomför enligt en planerad intervjuguide.

Utskrift/bearbetning: Det insamlade intervjumaterialet skall förberedas för analys, vilket innebär att eventuella inspelningar förs över till skriftspråk och renskrivning av anteckningar.

Analys: Här genomförs analys av den insamlade data. Lämplig metod för analysen beror på undersökningens syfte och ämne.

Verifiering: Här skall undersökningens validitet och reabilitet fastställas. Reabilitet hänför tillresultatets konsistens och validiteten till om studien undersöker vad den är avseddtill att göra.

Rapportering: Undersökningen renskrivs utifrån resultaten till en rapport enligt vetenskapliga kriterier.

3.1.4.2 Genomförande

Undersökningens syfte är att ta reda på om TDD har gett de fördelar som det sägs i teorin och om utvecklarna anser att metoden uppfyller de förväntningar som Jordbruksverket har på metoden.

Undersökningen kommer att genomföras på utvecklare inom projektteamet. Utifrån tematiseringen gjordes två frågeguider. Frågorna rörde hur utvecklarna såg på produktivitet och kodkvalité med den nya metoden och vilka faktorer som kunde påverka detta. En guide med frågor anpassade för utvecklare och en annan för testplanerare och teststrateg. Detta skickades sedan ut till deltagarna tillsammans med Health check, så de hade möjlighet att förbereda sig inför intervjun.

Intervjuerna bestod av samtliga sex personer, varav fyra utvecklare som har programmerat enligt TDD i projektet och två personer som har arbetat med tester enligt metoden. Intervjuerna pågick under 30 minuter, där frågor ställdes utifrån frågeguiden. Intervjupersonen fick därefter tala fritt kring frågorna, och följdfrågor ställdes vid behov. Anteckningar gjordes under hela intervjun.

Efter intervjuerna renskrevs alla anteckningar och resultatet sammanställdes i tabellform där svaren rangordnades efter kategorierna produktivitet, kodkvalité och effektivitet, därefter vilken faktor som påverkade kategorin och om det verkade positivt eller negativ för kategorin. Detta gjordes för att underlätta analysen i ett senare steg.

Deltagare 1:

Roll i projektet: Utvecklare
Är konsult med mer än 10 års erfarenhet av utveckling, och har tidigare erfarenhet av TDD från flera olika projekt.

Deltagare 2:

Roll i projektet: Utvecklare
Har cirka 15 års erfarenhet av utveckling, men ingen tidigare erfarenhet av TDD. Är också den person som har haft mest kunskap om hur tidigare Kundsystem var uppbyggt.

Deltagare 3:

Roll i projektet: Utvecklare
Har cirka 12 års erfarenhet av utveckling, men ingen tidigare erfarenhet av TDD.

Deltagare 4:

Roll i projektet: Utvecklare
Är konsult med 10 års erfarenhet av utveckling, och har tidigare erfarenhet av TDD från flera tidigare projekt.

Deltagare 5:

Roll i projektet: Testledare
Är konsult och har cirka 1,5 års erfarenhet av att arbeta med tester, men har inte arbetat med TDD tidigare. Huvuduppgiften har varit att veta vad för funktionalitet som skall testat och skapa en plan för detta.

Deltagare 6:

Roll i projektet: Teststrateg
Har mer än 20 års erfarenhet av att arbeta med tester, inte lika länge som teststrateg. Har tidigare erfarenhet av TDD som utvecklare, men ingen direkt som teststrateg.

4 Resultat och analys

I detta kapitel kommer resultatet från den teoretiska undersökningen, samt de metodiska tillvägagångssätten.

4.1 Delfråga 1

Vilka är de mest framträdande egenskaper som kan identifieras i samband med användande av TDD i utvecklingsprocessen?

De flest framträdande för- och nackdelar som har förekommit tillsammans med de undersökningar som har granskats är bland annat:

- Ökad feedback till utvecklare
- Uppgiftsorienterad/Nedbrytbarhet
- Kvalitétsökning
- Låg nivå på design som hjälper till vid namnsättning
- Programförståelse
- Ökad effektivitet
- Minskat arbete för små åtgärder

Nackdelarna är:

- Komplicerad tillämpning
- Stor tillit för refraktionering
- Otillräcklig kompetens

Nedan presenteras resultaten av de fem tidigare undersökningarna.

Författare	Typ av studie	Deltagare (antal)	Utvecklingsmetod	Kvalitet	Produktivitet
Hagner och Müller	Slutet experiment	Studenter	TDD	Ingen skillnad	Ingen skillnad
Maximilien, Williams och Vouk	Fallstudie	Prof. programmerare	TDD	TDD är bättre	Ingen skillnad
George och Williams	Slutet experiment	Prof. programmerare	TDD	TDD är bättre	TDD är sämre
Erdogmus, Morisio och Torchiano	Slutet experiment	Studenter	TDD	Ingen skillnad	TDD är bättre
Janzen och Saiedien	Slutet experiment	Studenter	TDD	Ingen skillnad	TDD är bättre

Fig 2. En tabell som sammanställer resultat från fem olika nämnda undersökningar.

4.2 Delfråga 2

Vilka egenskaper/förbättringar förväntar sig Jordbruksverket från TDD?

4.2.1 Resultat

Relevant data har hittats från två olika källor, Health check, och metoddokumentation

4.2.1.1 Health Check [15]

Rapporten resulterade i flera svagheter som Jordbruksverket borde åtgärda för att optimera sitt testarbete. De flesta punkterna rörde förbättringsarbete kring test och kvalitetsarbetet i helhet, så som testledning, testprocess, infrastruktur, förberedelser och genomförande. Detta resultat visar de punkter som direkt eller indirekt borde åtgärdas genom att använda TDD som utvecklingsmetod.

- Ett av de främsta problemen är att test kommer sent in i processen. Ofta skall det ske snabbt och tätt inpå driftsättning, vilket leder till tidspress.
- Vid tidspress prioriteras ofta funktionalitet framför kvalité vilket leder till en dyr förvaltning.
- Defekter hittas för sent på grund av oklara krav och otillräckliga tester
- Utvecklingsmetoden följs inte heller till fullo av alla och det gör att kvalitén kan variera. Metoden är inte helt uppdaterad vilket gör det svårt att veta exakt hur metoden skall följas.
- Jordbruksverket IT-system är väldigt komplexa som gör det svårt att testa helheten då många delar måste samverka. Samtidigt gör komplexiditeten det svårt att bryta ner funktionalitet så de blir testbara.
- Idag saknas det någon ansvarig för att helheten testas, då vill säga att förändringar i applikationer och strukturella förändringar testas.
- Testtänket uppfattas inte vara tillräckligt brett och det saknas bevis om tester har genomförts. I nuläget kan funktionalitet undantas från test på grund av tron att förändringen inte påverkan existerande funktionalitet plus att vetskapen om hur något skall testat saknas.
- I nuläget finns inget verktyg som hanterar testdokumentation, vilket gör det svårt att hitta artefakter att återanvända.
- Inga automatiserade tester utförs i dagsläget.

4.2.1.2 Metoddokumentation [4]

Jordbruksverket har flera regler för systemutveckling som skall användas vid tillämpning av metoden på Jordbruksverket.

TDD skall tillämpas vid all utveckling och testerna skall driva programmeringen. Detta skall ske enligt den standardiserade modellen av TDD. Genom att tillämpa TDD förväntas följande egenskaper enligt dokumentation.

- Buggar hittas tidigare i utvecklingen.
- Automatiserade tester underlättar arbetet med refraktionering.
- Ökad kvalitet genom fokus på test.
- Skapar en heltäckande testsvit för programmet, som kan underlätta förvaltning.
- Fokus läggs på funktionalitet som verkligen behövs.
- Ger utvecklaren återkoppling på kodens kvalité.
- Testfallen beskriver vad funktionaliteten skall uträtta, vilket ökar utvecklarens förståelse för kraven.
- Utvecklingstiden minskar.

4.2.2 Analys

Genom granskning av resultaten från både Health check och dokumentation går det att lyfta ut några punkter som kan ses som primära förväntningar på metoden som vidare analys kan utföras på.

- Test skall komma in tidigare i processen.
- Buggar skall framkomma tidigare i processen.
- Komplexiditeten bör minska då tester skall underlätta nedbrytning av funktionalitet.
- Utvecklaren får den feedback denne behöver för att producera kod med god kvalitet.
- Genom att lägga fokus på test skall kvalitén öka.
- Automatiserade tester skall bidra till att förbättra kodkvaliten
- Förvaltning skall bli smidigare i och med att en testsvit har byggts upp kring programmet under utvecklingen.
- Förståelsen för både krav och programkod skall öka.
- Utvecklingstiden minskar.

Ett av Jordbruksverket främsta problem är att test kommer sent in i utvecklingsprocessen, vilket skapar en tidspress som kan försämra kvalitén och leda till dyr förvaltning. George B. och Williams L [13] nämner i sin undersökning att buggar skall kunna åtgärdas i ett tidigt skede, och kan därigenom minska kostnader och tid för utveckling. Detta tyder på att Jordbruksverkets problem med sena tester kan lösas och bidra till att buggar hittas tidigare och ge en mer sparsam förvaltning.

Erdogmus H, Morisio M, Torchiano M [12] beskriver i sin undersökning möjligheten till att kunna bryta ner funktionaliteten genom att mocka bort dem, och kan på så sätt testa varje objekt var för sig. Detta gör att komplexiditeten minskar och underlättar tester av helheten. Eftersom Jordbruksverkets system är väldigt komplexa utgör detta en viktig egenskap för att kunna förbättra testarbetet.

Förutom nedbrytbarhet nämner också Erdogmus H., Morisio M., Torchiano M. [12]. att TDD ger utvecklaren feedback i sin kodning och verifierar att implementerad kod inte påverkar existerande kod. Eftersom alla enhetstester måste passera testerna ger det också en kvalitetsstämpel på koden. Detta skulle hjälpa Jordbruksverkets utvecklare att förbättra sitt testtänk i och med att enhetstesterna verifierar att koden inte påverkar existerande funktionalitet, samt att all kod som är implementerad är ett bevis på att den är testad. Genom enhetstester måste även utvecklaren tänka igenom hur kraven återspeglas i testerna vilket bidrar till att all funktionalitet täcks.

I och med införandet av TDD som utvecklingsmetod kommer automatiserade tester att införas. I Maximilien M., Williams L. och Vouk M. [14] undersökning visades en reducering på 40 % av defekterna jämfört med traditionella metoder. De ansåg också att automatiserade enhetstesterna byggde upp en testsvit kring programmet som senare blev en tillgång som kunde förbättras med tiden. Detta tyder på att testerna skulle hjälpa Jordbruksverket genom att både förbättra kodkvaliten och att underlätta förvaltningen genom att utnyttja den existerande testsviten.

I nuläget saknas det verktyg som kan hantera testdokumentation, vilket försvårar återanvändning av artefakter. I den undersökning som Hagner O. och Müller M. [10] gjorde såg de att den grupp som använde TDD kunde återanvända programmetoder mer effektivt. Detta såg de som ett bevis på att programförståelsen ökade och att utvecklarna kunde lättare se hur metoderna användes.

Utifrån de undersökningar som granskades har inte ett entydigt resultat framkommit att utvecklingstiden skall minska. Endast två av de fem undersökningarna ansåg att utvecklingstiden hade minskat, men det fanns inget som kunde tyda på ett sammanhang i resultatet mellan de två. Erdogmus H, Morisio M, Torchiano M. drog sin slutsats ifrån att utvecklare skriver fler tester per enhet och att det då leder till proportionellt ökad produktivitet. Janzen och Saiden stödde sin slutsats i att utvecklarna lade ner mindre tid än de i andra testgrupperna. Även om det finns vissa bevis som pekar på att utvecklingstiden kan minska är det ingen självklar egenskap som går att förvänta vid införande.

4.3 Delfråga 3

Finns det stöd för att TDD klarar av att motsvara de krav som Jordbruksverket har satt upp, samt går det att påvisa att metoden kan leverera de teoretiska fördelar som identifieras?

I föregående avsnitt analyserades flera primära egenskaper hos TDD som antas av Jordbruksverket att vara förväntade vid införandet. För att kunna svara på ovanstående frågeställning kommer analysen av intervjuerna att grunda sig på dessa egenskaper och hur dessa i praktiken uppfyller angivna förväntningar.

De förväntningar som har analyserats i tidigare stycke var:

- Test skall komma in tidigare i processen.
- Buggar skall framkomma tidigare i processen.
- Komplexiditeten bör minska då tester skall underlätta nedbrytning av funktionalitet.
- Utvecklaren får den feedback denne behöver för att producera kod med god kvalitet.
- Genom att lägga fokus på test skall kvalitén öka.
- Automatiserade tester skall bidra till att förbättra kodkvaliten
- Förvaltning skall bli smidigare i och med att en testsvit har byggts upp kring programmet under utvecklingen.
- Förståelsen för både krav och programkod skall öka.
- Utvecklingstiden minskar.

4.3.1 Intervjuresultat

Resultaten från intervjuerna har rangordnats efter produktivitet och kodkvalitét, samt vilka faktorer som har nämnts kring dessa.

4.3.1.1 Produktivitet

Under intervjun ställdes frågor om hur de upplevde produktiviteten med metoden. Dessa innefattade frågor kring nerbrytbarhet, kodförståelse och tester.

Samtliga av utvecklarna ansåg att TDD ökade kodförståelsen och gav beskrivande namnsättningar(P1). 50 % nämnde att förvaltningen borde förbättras och bli smidigare(P2). 75 % ansåg att refraktionering underlättade kodningen(P3).

75 % av de tillfrågade ansåg att testerna underlättade nedbrytning av kod(P4). Samliga nämnde att enhetstesterna gav dem en direkt verifiering av kodändringar(P5). Av de mer negativa faktorerna ansåg samtliga att metoden kräver både övning och erfarenhet för att uppnå full produktivitet(N1). Sist ansåg 75 % att komplexiditeten var ett hinder i utvecklingen(N2).

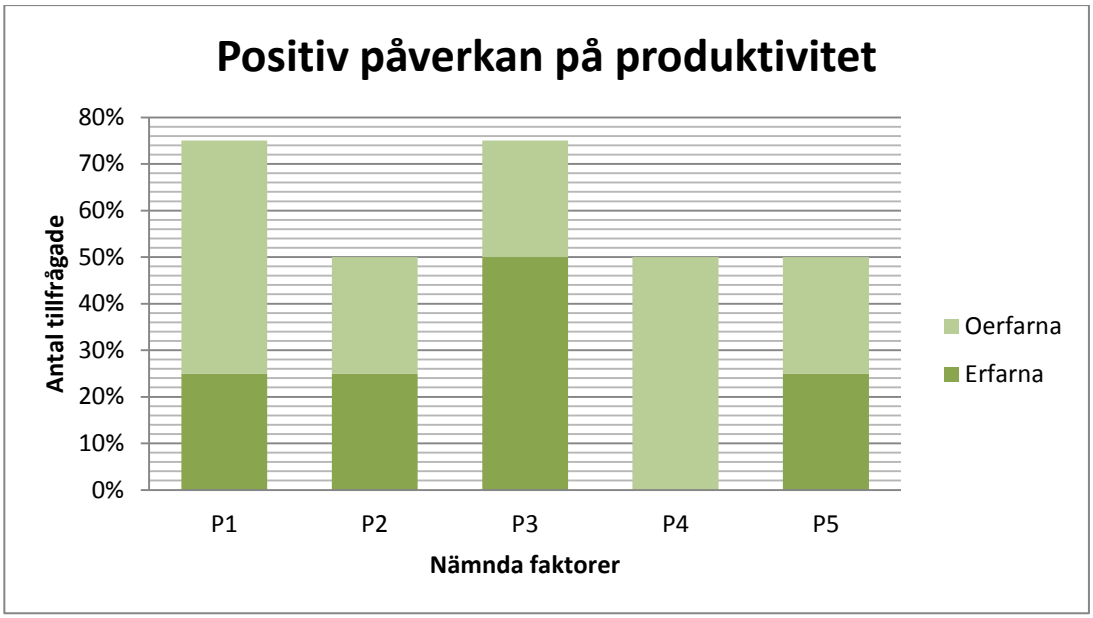

Fig 3. Tabell 1:1, visar faktorer med positiv påverkan på produktivitet

P 1: Ökad kodförståelse
P 2: Förvaltning underlättas
P 3: Refraktionering underlättar utveckling
P 4: Ökad nedbrytbarhet
P 5: Enhetstester ger feedback

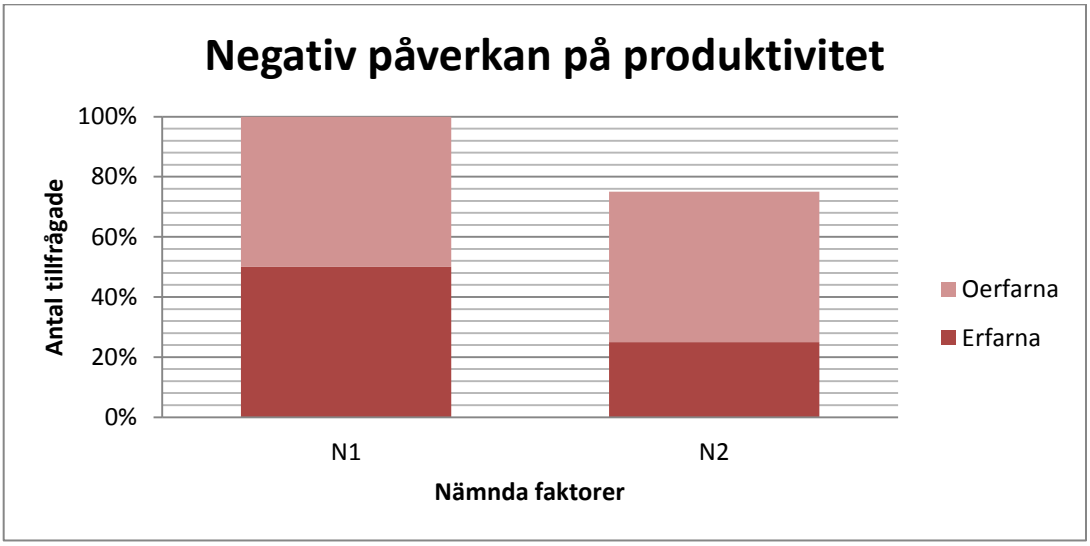

Fig 4. Tabell 1:2, visar faktorer med negativ påverkan på produktivitet

N 1: Kräver övning och erfarenhet
N 2: Komplexiditet

4.3.1.2 Kodkvalité

Frågor kring kodkvalité ställdes till både utvecklare och de som arbetade med planering och verkställande av tester, för att få åsikter från båda sidorna.

Genom att skriva testerna först uppkom flera fördelar men också nackdelar som de tillfrågade såg. 50 % nämnde att buggar hittas tidigare(P1).

33 % kände att de kunde lämna ifrån sig snyggare kod(P2).

50 % ansåg att metoden gav utvecklaren möjligheten att kunna optimera sin kod(P3) och 33 % tyckte att enhets och automatiserade tester förbättrade kvalitén(P4).

Cirka 16 % nämnde att mer funktionalitet täcks av tester(P5). 16 % ansåg också att mockning var en fördel för kvalitéten(P6).

Av de negativa faktorerna nämndes komplexitet av 66 % (N1).

50 % nämnde att det krävdes struktur för att skriva bra tester(N2) och 50 % tyckte att dålig erfaren ledde till slarv i metoden(N3).

33 % Såg en nackdel i att förlita sig så mycket på tester som det görs i TDD(N4) och 50 % nämnde att det saknades lämpliga verktyg för kontroll av testtäckning(N5).

Slutligen såg 16 % en risk i att refraktionering inte är tvingande för utvecklare.

Som ett avslut på intervjuerna ställdes frågan om de tyckte att TDD var en lämplig utvecklingsmetod för IT-avdelning. Där ansåg 100 % att TDD var en lämplig metod.

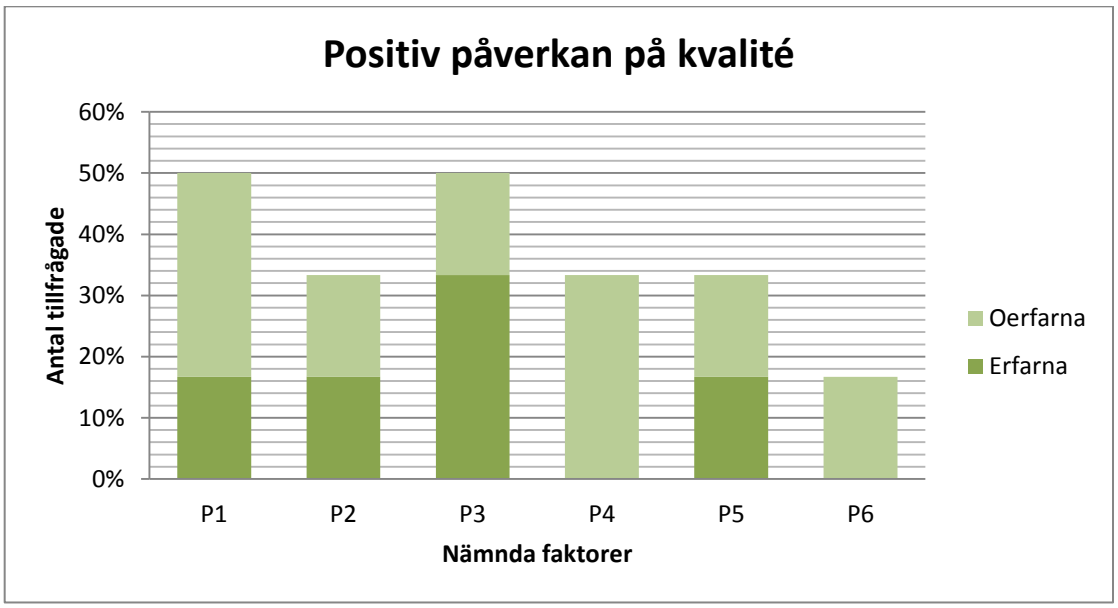

Fig 5. Tabell 2:1, visar faktorer med positiv påverkan på kvalité

P1: Buggar hittas tidigare
P2: Utvecklare lämnar ifrån sig snyggare kod
P3: Uppmuntrar till optimering av kod
P4: Enhets/automatiserade tester ger bättre kvalité
P5: Mer funktionalitet testas
P6: Mockning kan förbättra kodkvalité

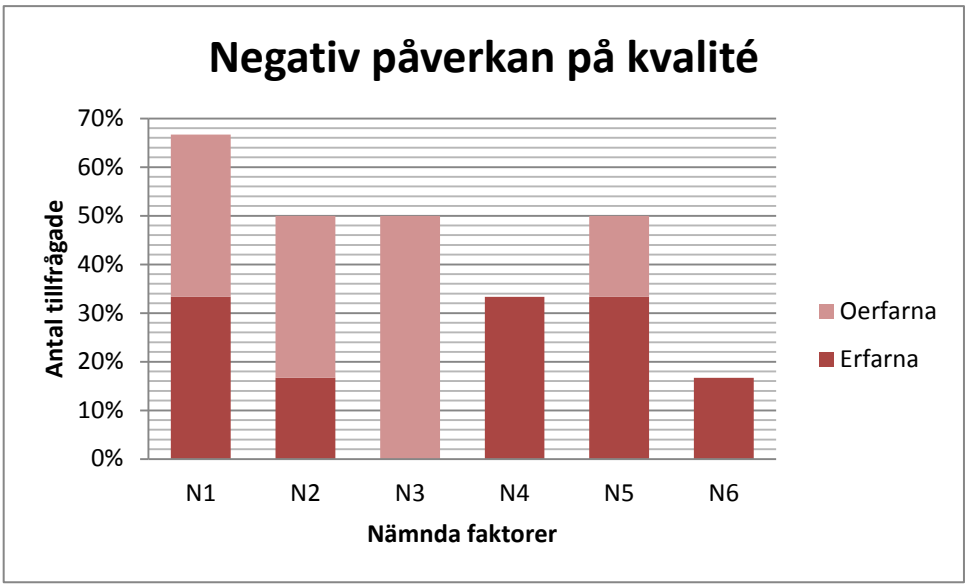

Fig 6. Tabell 2:2, visar faktorer med negativ påverkan på kvalité

N1: Komplexiditet
N2: Kräver god struktur för tester
N3: Dålig erfarenhet leder till att allt inte testas korrekt
N4: Stor tillit till test
N5: Avsaknad av lämpliga verktyg för test
N6: Refraktionering är inte tvingande

4.3.2 Analys

Genom sammanställningen av intervjuerna gick det att se att det finns några tydliga för och nackdelar med TDD.

4.3.2.1 Produktivitet

De nämnda faktorerna som skulle kunna påverka produktiviteten visas i tabell 1:1 och 1:2

Av utvecklarna ansåg samtliga att TDD skapade en mer lättförståelig kod(P1). Detta berodde på att de kunde se vilken funktionalitet som testades i en specifik kodklass. Det fanns också antydningar till att metoden hjälpte till att upprätthålla en bra namnstandard på koden. Detta stämmer överens med vad Hagner O. och Müller M. [10] såg i sin undersökning, där utvecklarna i deras testgrupp för TDD återanvände fler metoder än övriga testgrupper.

Flertalet av utvecklarna tror också att förvaltningen skulle komma att bli mer smidig(P2) genom att en testsvit har byggs upp kring systemet under projektets gång. Detta leder till att inte lika mycket tid behöver läggas på att skriva nya tester. Detta stämmer med Maximilien M., Williams L. och Vouk M. [14] som också såg att den testsvit som hade skapats kunde ses som en tillgång i fortsatt förvaltning.

Utvecklarna såg också refraktionering som något positivt(P3). Detta berodde på att testerna gjorde dem mer säkra i att gå tillbaka och göra ändringar i sin kod, istället för att ta tid och arbeta runt befintlig kod för att hitta en lösning. Erdogmus H., Morisio M., Torchiano M. [12] såg också detta som en egenskap inför deras studie. Genom att varje kod måste passera alla tester innan implementation såg de att koden fick en kvalitetsstämpel.

De upplevde också att det var lättare att dela upp koden till mer hanterbara delar med TDD(P4). Detta berodde på att testerna ansågs hjälpa till att bryta ner funktionaliteten, främst via mockning som omnämns senare kring kvalitét. Detta gick att finna stöd i Erdogmus H, Morisio M, Torchiano M.s undersökning som beskrev att testerna hjälpte till att bryta ner funktionaliteten [12]. Används mockning så underlättades detta test av enskilda objekt. Genom nedbrytning av koden leder detta indirekt till P1 som gör koden med lättförståelig eftersom det blir mindre kod att gå igenom. Hagner O. och Müller M. [10] såg i sin studie att kodförståelsen hade ökat genom att utvecklarna återanvände kod mer effektivt.

Utvecklarna tyckte också att TDD underlättade kodningen genom att enhetstester gav dem feedback direkt istället för att de måste bekräfta genom GUI varje gång en ändring hade gjorts(P5). Detta bekräftar Erdogmus H., Morisio M., Torchiano M. [12] genom att utvecklaren fick en verifiering om ändringar i koden har implementerats eller inte.

I tabell 1:2 visas faktorer som har negativ påverkan på produktiviteten.

En av faktorerna som nämndes var att metoden krävde mycket av dem gällande övning och erfarenhet(N1) för att de skulle uppnå den efterfrågade produktiviteten. Detta var mer förekommande hos de tillfrågade som saknade tidigare erfarenhet av TDD, än de med erfarenhet. Vilket ger förklaringen till att de inte kunde anse sig mer produktiva i nuläget. Precis som i de undersökningar som har granskats så är det även i detta fall svårt att bedöma om utvecklingstiden verkligen minskar. I detta fall beror det mycket på utvecklarens erfarenhet i att tillämpa metoden, samt att det kan vara svårt att se en minskad tidsåtgång under en kort period.

Slutligen ansåg utvecklarna att komplexiditeten i systemet kunde påverka produktiviteten på grund av att det är flera system som skall samverka(N2) och det krävde en större arbetsinsats för att skriva kod och tester för dessa. Detta beskrivs av George B. och Williams L [13] som ser praktiskt tillämpning av mockning som någonting som kan kräva mycket tid av utvecklaren.

4.3.2.2 Kvalitét

I tabell 2:1 och 2:2 beskrivs de faktorer som påverkar kodkvalitén.

En av de mest nämnda faktorerna (främst av utvecklare) var att de ansåg att kvalitén ansågs öka genom att testerna driver fram buggar(P1). Om utvecklaren upptäcker genom enhetstesterna buggar på ett tidigt stadium så höjs kvalitén. Detta styrker Erdogmus H., Morisio M., Torchiano M. [12] som tidigare har nämnts genom att utvecklaren får en bekräftelse på att koden har implementerats, och får då samtidigt en kvalitetsstämpel.

Främst utvecklarna ansåg också att de kunde leverera snyggare kod med högre kvalitét än innan beroende på att tester sker löpande i utvecklingen(P2). Detta stämmer med vad Erdogmus H., Morisio M., Torchiano M. [12] nämner ovan då utvecklaren får en kvalitetsstämpel på sin kod och gör denne medveten om att all kod som produceras håller god kvalitet.

De tillfrågade ansåg att testerna uppmuntrade utvecklare att våga ändra i sin kod till förbättring, både kvalitéts- och prestandamässigt(P3). Denna faktor har också ett beroende till P4 där de tillfrågade såg att genom att köra enhets- och automatiserade tester ges bättre kvalité. Detta berodde på att de såg att testerna även kunde fånga upp buggar som kommer från påverkande tjänster. Detta kan styrkas i Maximilien M., Williams L. och Vouk M. [14] studie som visade en reducering på 40 % av defekterna jämfört med traditionella metoder genom att använda enhets- och automatiserade tester. Samtidigt nämner också George B. och Williams L [13] att genom att buggar hittas tidigare i processen går det att spara tid för utveckling.

Genom testerna kunde de också se att det är mer funktionalitet som testas jämfört med tidigare metod(P5). I och med att enhetstester skrivs för varje funktionalitet leder det i längden till att mer funktionalitet täcks av tester.

Mockning som tidigare förekom som en negativ påverkande faktor för produktivitet nämns här också fast i den formen att mockning påverkar kvalitéten genom att det går att koppla bort funktionalitet från varandra och göra dem till enskilda testbara enheter(P6). Denna uppfattning går att finna stöd från Erdogmus H, Morisio M, Torchiano M [12] som anser att testerna kan bryta ner funktionalitet i mindre mer hanterbara uppgifter.

I tabell 2:2 visas de negativa aspekterna på kvalitéten som uppkom.

Även här nämndes systemets komplexitet som en negativ faktor på kvalitén (N1). Det berodde bland annat på att utvecklarna ansåg att det var svårt att skriva testfall när flera system skulle samspela och ju närmare GUI de kom. De som arbetade med test upplevde att planeringen av tester försvårades. Som angetts tidigare ansåg George B. och Williams L [13] bland annat att mer komplicerade fall av mockning kan komma och bli en större nackdel än fördel i vissa fall då det krävs mycket arbete att skriva tester.

Främst utvecklarna ansåg att det krävdes mycket struktur för att skriva tester(N2). Flera av dem, främst oerfarna upplevde att det var svårt att veta omfattningen av tester och saknade en bekräftelse på att funktionalitet som de testar verkligen är rätt. Detta ledde indirekt till resultatet i N3 där de oerfarna utvecklarna tyckte att de inte bidrog till ökad kodkvalité. De ansåg att det berodde på att de i brist på erfarenhet inte kunde skriva alla tester fullt ut. Även här stärks denna uppfattning av George B. och Williams L [13] som nämner utvecklarens kompetens som något avgörande för att kunna skriva rätt tester för rätt funktionalitet.

George B. och Williams L [13] såg också att den stora tilliten till refraktionering som finns inom TDD som något negativ, eftersom den testsvit som finns behöver konstant underhåll för att upprätthålla kvalitén. Detta sågs också bland testarna som ett bekymmer (N4). De såg att om de automatiserade testerna inte underhölls

på ett bra sätt kunde det utgöra en risk för kvalitén. Vidare nämndes också att det saknades lämpliga verktyg för att kunna kontrollera testtäckningen.(N5).

En aspekt som inte tycks ha nämnts tidigare i teorin är att refraktionering inte är tvingande för en utvecklare att genomgå(N6), vilket kan leda till att utvecklaren kan välja att inte optimera sin kod som i sin tur kan leda till försämrad kvalitét.. Denna uppfattning har inte stöd i någon funnen teori, men hör till en problematik som är värd att notera till vidare slutsats.

5 Diskussion och slutsatser

5.1 Resultatdiskussion

Syftet med examensarbetet var att undersöka hur lämplig TDD är som utvecklingsmetod för IT-avdelningen. För att kunna ta reda detta har den primära frågeställningen brutits ner till tre delfrågeställningar.

Delfråga 1: Vilka är de mest framträdande egenskaper som kan identifieras i samband med användande av TDD i utvecklingsprocessen?

Syftet med frågeställningen har varit att ta fram de möjliga positiva och negativa egenskaper som TDD antas föra med sig vid ett införande av metoden. Genom att använda andra praktiska studier gick det att verifiera att dessa egenskaper även kunde existera i praktiken. Resultatet från dessa studier kunde jag sedan vid nästa frågeställning använda till vidare analys och bekräfta senare ställda frågeställningar. Genom detta anser jag att min teoretiska bakgrund har kunnat uppfylla ovanstående syfte

Delfråga 2: Vilka egenskaper/förbättringar förväntar sig Jordbruksverket från TDD?

Syftet med delfråga två har varit att hitta de förväntningar som Jordbruksverket har på metoden vid ett införande. Utifrån resultatet valdes primära förväntningar ut och dessa har sedan kunnat bekräftas mot resultatet från ovanstående frågeställning, och kan därigenom ses som realistiska för projektet.

Jag anser att efter genomförd analys av nämnda förväntningar att jag har kunnat skapa en bild av de förväntningar och förbättringar som Jordbruksverket hoppas på vid ett införande av TDD.

Delfråga 3: Finns det stöd för att TDD klarar av att motsvara de krav som Jordbruksverket har satt upp, samt går det att påvisa att metoden kan leverera de praktiska fördelar som den påstås medföra?

Syftet med intervjuerna var att ta reda på om metoden gav de fördelar som Jordbruksverket förväntade. Genom intervjuerna gick det att testa de resultat som hade framkommit genom analysen mot de egenskaper som önskades. Det resultat som jag fick fram har varit lovande och har senare lett till en analys som kan visa på att flera av de positiva egenskaper som beskrevs i teorin även kan existera i praktiken.

Något som framkom under intervjuerna som var aningen oväntat var att den generella uppfattningen av metoden var olika och att så många negativa aspekter kom till ytan som inte tidigare var omnämnda i teorin. Till exempel att skriva enhetstester först eller att refraktionering inte är en obligatorisk del i utvecklingsprocessen, vilket de framstod som i teorin. Främst var det oerfarna utvecklare som såg fler hinder och kände en stor tröskel inför införandet, men

även de mer erfarna talade om den långa vägen de haft till den kompetens de har idag.

Slutligen ansåg de som deltog i intervjun att metoden kan förbättra Jordbruksverkets testarbete genom att test får ett större utrymme och kommer tidigare in i processen, och tillsammans med rätt införande, utbildning och rätt verktyg kan TDD vara en bra utvecklingsmetod för IT-avdelningen. Utifrån resultatet går det att se att de förväntningar som Jordbruksverket har på metoden kan ses som realistiska.

Huvudfråga: Hur lämplig är den testdrivna metod som har valts att tillämpas för pilotprojektet ”Kund”, för systemutveckling inom Jordbruksverket?

I delfråga ett hade flera egenskaper, stärkta av empiriska studier framkommit, både negativa och positiva. Dessa har jag sedan kunna bekräfta i delfråga två genom att matcha dessa mot de förväntningar som Jordbruksverket har på metoden genom att undersöka dokumentation kring testarbete och regler kring systemutveckling.

Efter de utförda intervjuerna kunde jag se av resultatet att flertalet av de förväntade kvalitéerna kunde uppfyllas. Det som inte stämde överens med Jordbruksverkets förväntningar var att intervjupersonerna inte kunde se att utvecklingstiden minskade utan ökade istället. Detta var på grund av oerfarenhet och komplexiditet i metoden. De ansåg också at det var svårt att se en ökad kvalité under en så kort period som projektet har löpt över. Det de däremot var överens om var att TDD skulle kunna bli en lämplig metod för Jordbruksverket längre fram när kompetensen väl har utvecklats.

Utifrån de resultat som har frambringats genom delfrågorna anser jag att jag har kunnat uppfylla det syfte jag har haft med metoden och ge en bild av metodens lämplighet.

5.2 Metoddiskussion

I detta kapitel redovisas reabiliteten och validiteten av resultatet.

5.2.1 Reabilitet

Litteraturstudierna har grundas sig främst på tidigare empiriska studier inom ämnet. Dessa undersökningar har förekommit i flera andra sammanhang och personerna som utfört dessa anser jag vara trovärdiga.

I intervjustudien användes endast sex personer. För att få en bättre tillförlitlighet borde fler personer inkluderats, men på grund av projektgruppens storlek gick det inte att utöka.

Att återge effekter på produktiviteten och kodkvalitét har varit komplicerat. För att kunna mäta exakt effekt behövs instrument och verktyg, något som jag inte har haft tillgång till. Jag anser dock att denna metod kan vara bättre vid större projektgrupper med tiotals deltagare som kan ge ett specifikt mönster, men i smågrupper med knappt tio personer skulle det troligen ge många variationer, precis som det nämns i de empiriska studierna som har granskats.

Slutligen anser jag att mitt val av metoder har gett mig ett trovärdigt resultat utifrån att den teoretiska och den praktiska undersökningen visar på liknande resultat.

5.2.2 Validitet

De empiriska studier som har används undersökte både produktiviteten och kodkvalitén med TDD. Förutom detta användes vid utvecklingen även JavaScript, det samma språk som användes vid utvecklingen av KUND.

För att vara säker på att få lämpliga svar i intervjuerna grundade jag mina frågor på resultatet från litteraturstudierna och dokumentation från Jordbruksverket. Detta gjorde att jag kunde få en fingervisning om vilket sorts resultat som jag borde förvänta mig.

Möjligheten fanns att jag kunde ha kontaktat utvecklare från utomstående konsultfirmor som hade erfarenhet av TDD för att kunna utöka intervjuerna. Jag uteslöt detta eftersom deras svar inte skulle kunna återspegla de erfarenheter som berörda personer i KUND har fått tagit del av.

Genom att inte ställa samma frågor till de som arbetade med tester och de som utvecklade, anser jag att det har gett ett bättre och mer ärligt resultat. Detta är för att till exempel en testplanerare inte har någon vetskap om hur TDD påverkade kodningen men däremot har mer ingående kunskap om hur testerna påverkades.

Slutligen anser jag att mitt val av metoder har kunnat ge dt svar jag har behövt för att kunna svara på de frågeställningar jag har haft.

5.3 Slutsatser och rekommendationer

Genom att ha kombinerat tillgänglig teori och empiri har jag kunnat nå fram till flera slutsatser som får mig att se TDD som en lämplig utvecklingsmetod för Jordbruksverket.

- **Förvaltning borde underlättas och bli mer effektiv.**
  Genom att upprätthålla automatiserade tester, och arbeta med test genom hela processen ger detta möjligheten till att underlätta framtida förvaltning och göra den mer effektiv

- **Ger Jordbruksverket en framtidssäker IT-kompetens.**
  TDD är en väldigt krävande metod, men med rätt införande, stöd och utbildning tyder resultatet på att utvecklarna känner sig säkrare och stoltare över sin kods kvalité. Detta gör att Jordbruksverket får en god IT-kompetens framöver.

- **Flera egenskaper förekommer även i praktiken.**
  Mitt resultat visar även på att flera egenskaper som presenteras hos TDD även är förekommande vid praktiskt tillämpning. Det går därför att dra slutsatsen att de flesta av förväntningarna som Jordbruksverket har haft är uppnåeliga och TDD har möjlighet att rätta till flera av de punkter som behöver åtgärdas enligt resultatet från Heath Check.

- **Minskar inte utvecklingstiden.**
  Även om min rapport har visat flera fördelar med TDD har det inte gått att bevisa att utvecklingstiden minskar. Däremot visar resultatet att metoden har en upplärningskurva där utvecklingen tar längre till, men minskar i och med att kompetensen hos utvecklaren ökar.

- **Svårt att se en ökning av kvalité i nuläget**
  På grund av den upplärningskurva som finns bland de som utvecklar enligt metoden, går det inte att se några bevis i nuläget att kvalitén har ökat. Däremot går det att dra slutsaten att kvalitén borde öka när kurvan börjar rätas ut och utvecklarna är mer vana att arbeta med metoden.

Som slutsats anser jag att TDD har potential att uppfylla de förväntningar som Jordbruksverket har satt upp för sin utvecklingsmetod men att detta är starkt beroende av att utvecklarna får tid på sig att lära sig och skaffa sig erfarenhet av metoden. Min rekommendation är att arbeta vidare med kompetensutveckling och stöd för projekt och förvaltning som kommer att arbeta med TDD. Jag vill även poängtera vikten i att underhålla test, för att kvalitén skall räcka ända fram, men, med rätt införande, stöd och utbildning har jag inget tvivel om att TDD blir en lämplig utvecklingsmetod för Jordbruksverket.

Slutligen vill jag tillägga att detta endast är en enskild undersökning utifrån Jordbruksverkets förhållande till metoden. Det går att visa att flera egenskaper kan förekomma i praktiken, men det inte tillräckligt för att komma fram till en generell slutsats av att vissa specifika fördelar alltid går att räkna med. Det skulle behövas en mycket mer omfattande undersökning med mätningar under en längre period för att nå fram till en generell slutsats.

6 Referenser

[1] Statens jordbruksverk, ”Jordbruksverkets mål och medel,” 13 11 2011. [Online]. Available: http://www.sjv.se/omjordbruksverket/jordbruksverketsmalochmedel.4.5aec661121e2613852800010081.html.[Använd 13 11 2011].

[2] Statens jordbruksverk, ”Detta är Jordbruksverket,” [Online]. Available: http://www.jordbruksverket.se/omjordbruksverket.4.5aec661121e26138528000506.html. [Använd 06 05 2011].

[3] Statens jordbruksverk, *IT-strategi 2010-2013,* Statens jordbruksverk, 2010.

[4] C. Jansson och Y. Linderstam, ”Regler för systemutveckling,” Statens jordbruksverk, Jönköping, 2011.

[5] A. Sweden, ”Om Agile,” [Online]. Available: http://www.agilesweden.com/. [Använd 22 05 2011].

[6] D. Astels, Test-Driven Development: A Practical Guide, Prentice Hall , 2003.

[7] S. W. Ambler, Agile modeling: Effective practices for eXtreme programming and the unified process, New York: John Wiley & Sons Inc., 2002.

[8] N. Nagappan, E. M. Michael, T. Bhat och L. Williams, ”Realizing quality improvement through test driven development: results and experiences of four industrial teams,” 2008.

[9] D. S. Janzen, ”An Empirical Evaluation of the Impact of Test-Driven Development on Software Quality,” University of Kansas, 2006.

[10] M. M. Müller och O. Hagner, ”Experiment about Test-First programming,” University of Karlsruhe, Karlsruhe, 2002.

[11] K. Beck, Test-Driven Development by Example, Boston: Addison-Wesley, 2003.

[12] H. Erdogmus, M. Morisio och M. Torchiano, ”On the Effectiveness of the Test-First Approach to Programming.,” IEEE Transactions on Software Engineering, 2005.

[13] B. George och L. Williams, ”A structed experiment of test-driven development.,” Proc. ACM Symp. Applied Computing, 2003.

[14] L. Williams, M. Vouk och E. M. Michael, ”Test-Driven Development as a Defect-Reduction Practice,” 14th International Symposium on Software Reliability Engineering, 2003.

[15] A. Nilsson och A. Mourelatos, ”Health Check,” SQS, Jönköping, 2010.

[16] A. Lantz, Intervjumetodik, Lund: Studentlitteratur, 2007.

[17] J.-A. Kylén, Att få svar: intervjun enkät, observation, Stockholm: Bonnier utbildning, 2004.

[18] S. Kvale och S. Brinkmann, Den kvalitativa forskningsintervjun, Lund: Studentlitteratur, 2009.

[19] S. Kvale, eEn kvalitativa forskningsintervjun, Lund: Studentlittertur, 2009.

[20] E. M. Michael och L. Williams, ”Assessing Test-Driven Development at IBM,” Proc. Int'l Conf. Software Eng. (ICSE), 2003.

7 Bilagor

Bilaga 1 Intervjuguide för utveckling

Bilaga 2 Intervjuguide för test

Bilaga 3 Intervjuanteckningar utvecklare 1

Bilaga 4 Intervjuanteckningar utvecklare 2

Bilaga 5 Intervjuanteckningar utvecklare 3

Bilaga 6 Intervjuanteckningar utvecklare 4

Bilaga 7 Intervjuanteckningar testare 1

Bilaga 8 Intervjuanteckningar testare 2

Bilaga 1: Frågeguide utveckling

1. Intro

- Vad har du haft för roll i teamet?
- Vad har din roll inneburit?
- Hur många års erfarenhet har du av att arbeta med tester?
- Har du haft tidigare erfarenhet av TDD?

I flera undersökningar som har gjorts har visat att TDD skall höja produktiviteten, kodkvalitén och effektiviteten. Dessa resultat har hittills bara framkommit genom ett fåtal experiment, vilket gör det svårt att se om metoden verkligen ger de fördelar som det sägs i teorin.

2. Generellt

- Nu är ni nästan klara med projektet, hur har det gått att utveckla enligt testdrivenutveckling?

I undersökningarna sägs TDD göra utvecklarna med produktiva genom att de kan bryta ner stories till mindre delar och utveckla fortare och smidigare tack vare att skriva enhetstester först.

3. Produktivitet

- Är detta någonting som ni har märkt av när ni arbetade enligt TDD?
- Vilka faktorer tror du påverkade?
- Skulle du kunna säga att TDD har gjort dig till en mer effektiv utvecklare?

En nämnd fördel med en testdriven metod är att den skall öka kodkvalitén med hjälp an enhetstesterna.

4. Kodkvalitét & tester

- Har ni sett detta under utvecklingen?
- Vilka faktorer tror du påverkade?
- Ökar kodförståelsen med enhetstesterna?
- Anser du att genom att skriva testerna först har gett dig den feedback som behövs för att kunna hitta de flesta buggarna innan leverans?

Jordbruksverket har ju också en hel del förväntningar på metoden. Det mesta handlar om att förbättra testarbetet som är en bieffekt från resultatet i Heath Check. Några saker som nämns är att test kommer sent in i processen,-funktionalitet prioriteras före kvalité, samt komplexitet medför svåra tester.

5. Health Check

- Hur anser du att TDD har påverkar SJV testarbete med hänsyn till Health check?
- Anser du att TDD är en lämplig utvecklingsmetod för IT-avdelningen?

6. Övriga frågor/Nackdelar?
7. Andra synpunkter?

Bilaga 2: Frågeguide test

1. Intro

- Vad har du haft för roll i teamet?
- Vad har din roll inneburit?
- Hur många års erfarenhet har du av att arbeta med tester?
- Har du haft tidigare erfarenhet av TDD?

I flera undersökningar som gjorts har visat på att TDD skall höja produktiviteten kodkvalitén och effektiviteten. Dessa resultat har hittills bara framkommit genom ett fåtal experiment, vilket gör det svårt att se om metoden verkligen ger de fördelar som det sägs i teorin.

En nämnd fördel med en testdriven metod är att den skall öka kodkvalitén tillsammans med den långsiktiga med hjälp an enhetstesterna.

2. Kodkvalitet

- Har ni sett detta under utvecklingen?
- Hur anser du att testerna har förbättrat kvaliteten?

SJV har ju mycket komplexa system.

- Har det varit möjligt att testa majoriteten av funktionaliteten?
- Ha det funnits någon funktionalitet som har varit särskilt komplicerad?
- Skulle du kunna säga att majoriteten av funktionaliteten har ni lyckats att täcka med tester?

*Jordbruksverket har ju också en hel del förväntningar på metoden. Det mesta handlar om att förbättra testarbetet som är en bieffekt från resultatet i Heath Check. Några saker som nämns är att test kommer sent in i processen, funktio*nalitet prioriteras före kvalité*, samt komplexitet medför svåra tester.*

3. Health Check

- Hur anser du att TDD har påverkar SJV testarbete med hänsyn till Health check?
- Anser du att TDD uppfyller de förväntningar som SJV har på metoden?
- Anser du att TDD är en lämplig utvecklingsmetod för IT-avdelningen?

4. Övriga frågor/Nackdelar?

5. Andra synpunkter

Bilaga 3: Intervjuanteckningar utvecklare 1

1. Intro

- Vad har du haft för roll i teamet?
    a. *Utvecklare*
- Vad har din roll inneburit?
    a. *Utveckling, lite som en mentor för de andra*
- Hur många års erfarenhet har du av att arbeta med tester?
    a. *Mer än 10 års erfarenhet*
- Har du haft tidigare erfarenhet av TDD?
    a. *Ja, ifrån olika projekt(är konsult)*

I flera undersökningar som gjorts har visat på att TDD skall höja produktiviteten, kodkvalitén och effektiviteten. Dessa resultat har hittills bara framkommit genom ett fåtal experiment, vilket gör det svårt att se om metoden verkligen ger de fördelar som det sägs i teorin.

2. Generellt

- Nu är ni nästan klara med projektet, hur har det gått att utveckla enligt testdriven utveckling?
    a. *Bra, men det krävs mycket arbete av utvecklaren*
    b. *TDD är inte en metod som är lätt att tillämpa, det krävs övning tills det sitter.*

I undersökningarna sägs TDD göra utvecklarna med produktiva genom att de kan bryta ner stories till mindre delar och utveckla fortare och smidigare tack vare att skriva enhetstester först.

3. Produktivitet

- Är detta någonting som ni har märkt av när ni arbetade enligt TDD?
    a. *Ja*
- Vilka faktorer tror du påverkade?
    a. *Mycket beror på utvecklaren. Utvecklaren måste känna sitt utvecklingsverktyg för att kunna vara mer produktiv.*
    b. *Övning ger färdighet.*
    c. *T.ex. Att verifiera persnr, adress m.m som inmatningsfält är lättare att testa med enhetstester än att gå till GUI och testa (är mer omständligt att logga in starta upp server m.m) Det kortar ner utvecklingsprocessen betydligt.*

- Skulle du kunna säga att TDD har gjort dig till en mer effektiv utvecklare?
    a. *Ja*

En nämnd fördel med en testdriven metod är att den skall öka kodkvalitén med hjälp an enhetstesterna.

4. Kodkvalitét & tester

- Har ni sett detta under utvecklingen?

a. *Ja*

- Vilka faktorer tror du påverkade?
    a. *Tester tvingar utvecklaren att tänka annorlunda, och det testar sin affärslogik hela tiden.*
    b. *Refraktionering uppmuntrar utvecklaren att testa sin kod än gång till och man blir inte lika rädd för att ändra i den.*
    c. *Mycket svårt att tänka test först, återigen beror mycket på utvecklaren.*
    d. *Större integrationstester/funktionstester gör det mer komplicerat, och det blir mer komplicerar ju närmare GUI man kommer pga av samverkan med andra system.*

- Ökar kodförståelsen med enhetstesterna?
    a. *Enhetstesterna beskriver vad det testar och hur vilket gör det lättförståeligt om man kommer in i någons arbete*

- Anser du att genom att skriva testerna först har gett dig den feedback som behövs för att kunna hitta de flesta buggarna innan leverans?
    a. *Ja, det får utvecklaren att tänka om hela tiden.*

Jordbruksverket har en hel del förväntningar på metoden. Det mesta handlar om att förbättra testarbetet som är en bieffekt från resultatet i Heath Check. Några saker som nämns är att test kommer sent in i processen, funktionalitet prioriteras före kvalité, samt komplexitet medför svåra tester.

5. Health Check

- Hur anser du att TDD har påverkar SJV testarbete med hänsyn till Health check?
    a. *Hade ej läst genom den. Men anser att det borde förbättra.*
- Anser du att TDD är en lämplig utvecklingsmetod för IT-avdelningen?
    a. *Ja, det är det väl, men mycket beror på utvecklaren.*

6. Övriga frågor/Nackdelar?

a. *Det är en mycket krävande process och utvecklaren behöver mycket övning innan han/hon blir duktig på att tillämpa TDD. Detta kan leda till "slarv" vid tester.*
b. *Man är inte tvungen till att refraktionera vilket gör att koden kanske inte har den kvalitén som den borde.*

7. Andra synpunkter

a. *TDD är inte en magisk lösning på alla problem om det inte finns en tanke bakom.*
b. *Mer utbildning kan göra utvecklarna bättre => mer effektiva.*

Bilaga 4: Intervjuanteckningar utvecklare 2

1. Intro

- Vad har du haft för roll i teamet?
    a. *utvecklare*
- Vad har din roll inneburit?
    a. *Avveckling av gamla kundsystemet och programmering.*
- Hur många års erfarenhet har du av att arbeta med tester?
    a. *12 år*
- Har du haft tidigare erfarenhet av TDD?
    a. *Nej*

I flera undersökningar som har gjorts har visat att TDD skall höja produktiviteten, kodkvalitén och effektiviteten. Dessa resultat har hittills bara framkommit genom ett fåtal experiment, vilket gör det svårt att se om metoden verkligen ger de fördelar som det sägs i teorin.

2. Generellt

- Nu är ni nästan klara med projektet, hur har det gått att utveckla enligt testdrivenutveckling?
    a. *Det har varit en tröskel. Det kanske låter bra i teorin, men rent praktiskt är det svårt. Har skrivit testfallen i efterhand eftersom det är svårt att veta exakt vad och hur man skall skriva testfallen.*
    b. *Testerna gör så att man vågar ändra i koden och man behöver inte vara orolig för vad det kan påverka för testerna tar hand om det jobbet.*

I undersökningarna sägs TDD göra utvecklarna med produktiva genom att de kan bryta ner stories till mindre delar och utveckla fortare och smidigare tack vare att skriva enhetstester först.

3. Produktivitet

- Är detta någonting som ni har märkt av när ni arbetade enligt TDD?
    a. *Inte personligen pga liten erfarenhet, men mycket handlar om att metoden praktiseras på rätt sätt. Tex refraktionering hjälper utvecklaren att våga ändra i koden istället för att ta långa omvägar och skriva onödig extra kod i redan stora kodstycken.*

- Vilka faktorer tror du påverkade?
    a. *I förvaltningen kan utvecklingen vara smidigare då de kan förlita sig på projektets tidigare tester och vågar därför ändra mer i koden.*

    b. *Anser att man borde involvera testledaren tidigare i processen som även kan ha synpunkter på enhetstester för att utvecklarna skall veta vad och hur de skall testa. Är osäker på om det man testar är rätt. Då har man mer förståelse för acceptanskriterierna och gör det lättare att programmera.*

- Skulle du kunna säga att TDD har gjort dig till en mer effektiv utvecklare?
    a. *Tror att det ökar längre fram i kedjan då man har fler testfall skrivna att förlita sig på, men erfarenhet spelar in också.*

En nämnd fördel med en testdriven metod är att den skall öka kodkvalitén med hjälp an enhetstesterna.

4. Kodkvalitét & tester

- Har ni sett detta under utvecklingen?
    a. *Ja*
- Vilka faktorer tror du påverkade?
    a. *Testerna hjälper till och fångar upp fel om ev. ändringar skulle påverka annan funktionalitet.*
    b. *Men mycket beror på hur man tillämpar metoden, slarvar man blir det en belastning.*

- Ökar kodförståelsen med enhetstesterna?
    a. *Praktiseras metoden rätt så tvingar den fram modellering och bra namnsättning som gör det lättare att förstå.*

- Anser du att genom att skriva testerna först har gett dig den feedback som behövs för att kunna hitta de flesta buggarna innan leverans?
    a. *Ja absolut, det har räddat mig många gånger.*

Jordbruksverket har ju också en hel del förväntningar på metoden. Det mesta handlar om att förbättra testarbetet som är en bieffekt från resultatet ~~i~~ Heath Check. Några saker som nämns är att test kommer sent in i processen,- funktionalitet prioriteras före kvalité, samt komplexitet medför svåra tester.

5. Health Check

- Hur anser du att TDD har påverkar SJV testarbete med hänsyn till Health check?
    a. *TDD gör det lättare att hantera kravändringar senare i processen.*
    b. *Test kommer in tidigt i processen.*
    c. *Bättre kodkvalité*
- Anser du att TDD är en lämplig utvecklingsmetod för IT-avdelningen?
    a. *Ja med rätt införande är det bra men då måste alla vara med på tåget och någon sorts organisation måste byggas upp kring detta så metoden utförs på rätt sätt.*

6. Övriga frågor/Nackdelar?

a. *Är en tröskel*
b. *Tidskrävande*
c. *Gör man inte på rätt sätt så blir det lätt slarv.*

7. Andra synpunkter?

a. *Man blir en mer professionell utvecklare, och man blir bättre på att estimera.*

Bilaga 5: Intervjuanteckningar utvecklare 3

1. Intro

- Vad har du haft för roll i teamet?
    a. *utvecklare*
- Vad har din roll inneburit?
    a. *Främst ren utveckling men var också leveransansv.arig Har bäst koll på KUND-systemets uppg.*
- Hur många års erfarenhet har du av att arbeta med tester?
    a. *15 år*
- Har du haft tidigare erfarenhet av TDD?
    a. *Nej*

I flera undersökningar som har gjorts har visat att TDD skall höja produktiviteten, kodkvalitén och effektiviteten. Dessa resultat har hittills bara framkommit genom ett fåtal experiment, vilket gör det svårt att se om metoden verkligen ger de fördelar som det sägs i teorin.

2. Generellt

- Nu är ni nästan klara med projektet, hur har det gått att utveckla enligt testdrivenutveckling?
    a. *Jobbigt och svårt att komma in i ett testtänk.*
    b. *TDD är en tidskrävande metod.*

I undersökningarna sägs TDD göra utvecklarna med produktiva genom att de kan bryta ner stories till mindre delar och utveckla fortare och smidigare tack vare att skriva enhetstester först.

3. Produktivitet

- Är detta någonting som ni har märkt av när ni arbetade enligt TDD?
    a. *Nej*
- Vilka faktorer tror du påverkade?
    a. *Inte personligen, men om man har med övning och erfarenhet borde man bli mer produktiv.*
    b. *En tröskel som är svår att ta sig över.*

- Skulle du kunna säga att TDD har gjort dig till en mer effektiv utvecklare?
    a. *Har inte nått den nivån ännu, behöver mer övning*

En nämnd fördel med en testdriven metod är att den skall öka kodkvalitén med hjälp an enhetstesterna.

4. Kodkvalitét & tester

- Har ni sett detta under utvecklingen?
    a. *Både ja och nej*

- Vilka faktorer tror du påverkade?
    a. *Test tar fram buggarna tidigt*
    b. *Svårt att bedöma vad/hur som skall testas och hur de skall delas upp. Man vill lätt testa en hel process istället för att dela upp testerna.*
    c. *Ju närmare GUI man kommer desto svårare är det att skriva tester.*
    d. *pga. av dålig erfarenhet har testerna skrivit efteråt och då är det svårt att veta om man vet att man har testat allt.*

- Ökar kodförståelsen med enhetstesterna?
    a. *Eftersom koden delas upp genom enhetstesterna blir det mindre kodsnuttar som gör det lättare att förstå.*
    b. *Kodklasserna har beskrivande namn vilket gör det lättare att förstå koden.*

- Anser du att genom att skriva testerna först har gett dig den feedback som behövs för att kunna hitta de flesta buggarna innan leverans?
    a. *Nu har testerna skrivit efteråt, men JA på sätt och vis eftersom testerna blir ett facit över det man har gjort.*
    b. *Ibland är man tvungen att dubbelkolla i applikationen via GUI så att det verkligen fungerar pga. dålig testtäcking. Detta beror på brist på tid och lämpliga verktyg som kan underlätta.*

Jordbruksverket har ju också en hel del förväntningar på metoden. Det mesta handlar om att förbättra testarbetet som är en bieffekt från resultatet i Heath Check. Några saker som nämns är att test kommer sent in i processen,- funktionalitet prioriteras före kvalité, samt komplexitet medför svåra tester.

5. Health Check

- Hur anser du att TDD har påverkar SJV testarbete med hänsyn till Health check?
    a. *Vi är precis i början av användandet av TDD och det är svårt att se om det kommer att lösa så mycket som det sägs. Men med tanke på att test numera kommer in tidigt i processen bör TDD förbättra. Vi måste också se hur långt vi kan komma med tester inom en rimlig arbetsinsats.*

- Anser du att TDD är en lämplig utvecklingsmetod för IT-avdelningen?
    a. *Blir det en naturlig del i processen så bli det bra.*

6. Övriga frågor/Nackdelar?
    a. *Det är en tröskel och metoden är tidskrävande.*

7. Andra synpunkter?
    a. *Har bara fått en dagskurs. Pga av dålig erfarenhet leder det till fusk.*

Bilaga 6: Intervjuanteckningar utvecklare 4

1. Intro

- Vad har du haft för roll i teamet?
    a. *utvecklare*
- Vad har din roll inneburit?
    a. *Främst programmering, arbetar nu i projektet BEA*
- Hur många års erfarenhet har du av att arbeta med tester?
    a. *10 år*
- Har du haft tidigare erfarenhet av TDD?
    a. *Ja, ifrån tidigare projekt*

I flera undersökningar som har gjorts har visat att TDD skall höja produktiviteten, kodkvalitén och effektiviteten. Dessa resultat har hittills bara framkommit genom ett fåtal experiment, vilket gör det svårt att se om metoden verkligen ger de fördelar som det sägs i teorin.

2. Generellt

- Nu är ni nästan klara med projektet, hur har det gått att utveckla enligt testdrivenutveckling?
    a. *Går bra. Tycker om att arbeta testdriven och tycker det är roligt att jobba i ett projekt där man tillämpar det fullt ut.*
    b. *Tycker att man vinner tid genom att man får svar genom testerna.*
    c. *Metoden fungerar som en sorts planeringshjälp med tester och det tvingar utvecklaren att tänka test.*
    d. *Ramverket från SJV gör det komplicerat att skriva tester.*

I undersökningarna sägs TDD göra utvecklarna med produktiva genom att de kan bryta ner stories till mindre delar och utveckla fortare och smidigare tack vare att skriva enhetstester först.

3. Produktivitet

- Är detta någonting som ni har märkt av när ni arbetade enligt TDD?
    a. *Ja*
- Vilka faktorer tror du påverkade?
    a. *Man tvingas att dela upp koden i mindre mer hanterbara delar. Utvecklingen blir dock mer komplicerad ju mer system som skall samarbeta.*

- Skulle du kunna säga att TDD har gjort dig till en mer effektiv utvecklare?
    a. *Känner att man kan lämna ifrån sig en snyggare kod och det gör mig stoltare som utvecklare, så jag känner mig effektiv.*

En nämnd fördel med en testdriven metod är att den skall öka kodkvalitén med hjälp an enhetstesterna.

4. Kodkvalitét & tester

- Har ni sett detta under utvecklingen?
    a. *Ja*
- Vilka faktorer tror du påverkade?
    a. *Testerna upptäcker fel tidigt och hjälper till att bryta ner koden, vilket gör den mer lättförståelig.*
- Ökar kodförståelsen med enhetstesterna?
    a. *Den ökar, testerna hjälper till att beskriva koden.*

- Anser du att genom att skriva testerna först har gett dig den feedback som behövs för att kunna hitta de flesta buggarna innan leverans?
    a. *Känner att man kan lämna ifrån sig en snyggare kod och det gör mig stoltare som utvecklare.*

Jordbruksverket har ju också en hel del förväntningar på metoden. Det mesta handlar om att förbättra testarbetet som är en bieffekt från resultatet i Heath Check. Några saker som nämns är att test kommer sent in i processen,- funktionalitet prioriteras före kvalité, samt komplexitet medför svåra tester.

5. Health Check

8. Hur anser du att TDD har påverkar SJV testarbete med hänsyn till Health check?
    a. -
9. Anser du att TDD är en lämplig utvecklingsmetod för IT-avdelningen?
    a. *Tycker det är en jättebra metod, speciellt för förvaltning eftersom den har en testsvit som man kan förlita sig på.*
    b. *Man kan våga ändra mer i koden och veta att testerna tar hand om andra tjänster skulle påverkas.*

6. Övriga frågor/Nackdelar?

a. *Refraktionering kan leda till att testerna blir inaktuella om man omarbetar koden för mkt.*

7. Andra synpunkter?

a. *Involvera testledaren tidigare i processen för bättre styrning*

Bilaga 7: Intervjuanteckningar testare 1

1. Intro

- Vad har du haft för roll i teamet?
    a. *Teststrateg*
- Vad har din roll inneburit?
    a. -
- Hur många års erfarenhet har du av att arbeta med tester?
    a. *Sedan 1991*
- Har du haft tidigare erfarenhet av TDD?
    a. *Ingen direkt erfarenhet som teststrateg, men som utvecklare i tidigare projekt*.

I flera undersökningar som har gjorts har visat att TDD skall höja produktiviteten, kodkvalitén och effektiviteten. Dessa resultat har hittills bara framkommit genom ett fåtal experiment, vilket gör det svårt att se om metoden verkligen ger de fördelar som det sägs i teorin.

En nämnd fördel med en testdriven metod är att den skall öka kodkvalitén med hjälp an enhetstesterna.

2. Kodkvalitét

- Har ni sett detta under utvecklingen?
    a. –
- Hur anser du att testerna har förbättrat kvaliteten?
    a. *Eftersom fler tester skrivs, så finns det fler tester som täcker koden => testtätheten ökar.*
    b. *Kodtäckningen ökar, mer funktionalitet testas och det som inte testat synliggörs. => Mängden kod som testas ökar*

SJV har ju mycket komplexa system.

- Har det varit möjligt att testa majoriteten av funktionaliteten?
    a. -
- Ha det funnits någon funktionalitet som har varit särskilt komplicerad?
    a. -
- Skulle du kunna säga att majoriteten av funktionaliteten har ni lyckats att täcka med tester?
    a. -

Jordbruksverket har ju också en hel del förväntningar på metoden. Det mesta handlar om att förbättra testarbetet som är en bieffekt från resultatet ~~i~~ Heath Check. Några saker som nämns är att test kommer sent in i processen,- funktionalitet prioriteras före kvalité, samt komplexitet medför svåra tester.

3. Health Check

- Hur anser du att TDD har påverkar SJV testarbete med hänsyn till Health check?
    a. *Främst komponenttester som är viktiga,*
- Anser du att TDD uppfyller de förväntningar som SJV har på metoden?
    a. *TDD är en liten del i testarbetet och mycket arbete kvarstår. Bl.a annat har vi börjat jobba med riskbaserade tester något som inte finns med i Health check.*
- Anser du att TDD är en lämplig utvecklingsmetod för IT-avdelningen?
    a. *Det är väldigt viktigt med bra verktyg som kan stötta i utvecklingen. Mycket beror på det.*
    b. *Inte heller nödvändigt att bedriva TDD på alla projekt. Mindre komplexa projekt kanske som man inte behöver utför så mycket tester på.*

4. Övriga frågor/Nackdelar?

a. *Man måste strukturera sitt arbete för att kunna skriva tester.*
b. *Mycket tid kan läggas på miljöer och verktyg för att arbetet skall löpa på. I och med att man behöver skriva mer kod (test+kod) så kan det äta upp mkt tid för mindre projekt.*
c. *Om man inte underhåller och automatiserar tester som skall köras ganska regelbundet kan det vara en risk att bara förlita sig på test.*
d. *Oftast testar utvecklaren bara sina antaganden, utan att se till vad verksamheten avsåg testa.*

5. Andra synpunkter?

a. TDD är en liten del i ett pussel och löser inte allt.

Bilaga 8: Intervjuanteckningar testare 2

1. Intro

- Vad har du haft för roll i teamet?
    a. *Testledare*
- Vad har din roll inneburit?
    a. *Ett planeringsjobb, huvuduppgiften är att ligga steget före och veta vad som skall testas och se eventuella resurser som kan behövas inför tester och andra förberedelser och dokumentera detta i en testplan.*
- Hur många års erfarenhet har du av att arbeta med tester?
    a. *1,5 år*
- Har du haft tidigare erfarenhet av TDD?
    a. *Nej*

I flera undersökningar som har gjorts har visat att TDD skall höja produktiviteten, kodkvalitén och effektiviteten. Dessa resultat har hittills bara framkommit genom ett fåtal experiment, vilket gör det svårt att se om metoden verkligen ger de fördelar som det sägs i teorin.

En nämnd fördel med en testdriven metod är att den skall öka kodkvalitén med hjälp an enhetstesterna.

2. Kodkvalitét

- Har ni sett detta under utvecklingen?
    a. Ja
- Hur anser du att testerna har förbättrat kvaliteten?
    b. *Har inte jobbat på samma nivå som enhetstester, men har i uppgift att veta vilken kodtäckning det finns på integrationstesterna, vilket ökar.*
    c. *Statiska tester hjälper till att ge bra kvalité även senare.*

SJV har ju mycket komplexa system.

- Har det varit möjligt att testa majoriteten av funktionaliteten?
    a. *Ja*
- Ha det funnits någon funktionalitet som har varit särskilt komplicerad?
    a. *Tjänster som går mot bussen gör det komplext planeringsmässigt, för att det är många delar som skall sammarbeta i rätt tillfälle. Då är mockning en fördel då man kan bryta ner och testa tjänsterna enskilt innan de större testerna kommer. T.ex att hämta uppgifter från skatteverket (folkbokföringen), Deras testpersoner kanske inte är tillräckliga och då spelar det ingen roll hur mycket som testas i KUND*

Jordbruksverket har ju också en hel del förväntningar på metoden. Det mesta handlar om att förbättra testarbetet som är en bieffekt från resultatet i Heath Check. Några saker som nämns är att test kommer sent in i processen,- funktionalitet prioriteras före kvalité, samt komplexitet medför svåra tester.

3. Health Check

- Hur anser du att TDD har påverkar SJV testarbete med hänsyn till Health check?
    a. *Som en fördel. Test kommer in tidigt i processen och ger utrymme för fler tester.*
    b. *Tester driver fram buggar i ett tidigt stadium.*
- Anser du att TDD uppfyller de förväntningar som SJV har på metoden?
    a. *Ja, test är med i hela processen, men samtidigt är det svårt att se alla fördelar på kort sikt.*
- Anser du att TDD är en lämplig utvecklingsmetod för IT-avdelningen?
    a. *Ja*

4. Övriga frågor/Nackdelar?

a. *Det är svårt och metoden tar tid att bemästra, men samtidigt ger det utrymme för att man kan utvecklas*

5. Andra synpunkter

a. -