GÖTEBORGS UNIVERSITET

Testarbete inom utveckling och förändring av integrationsplattformar

Testing activities at the stages of development and change regarding integration brokers

Andreij Abréu
Erik Öhman

Kandidatuppsats i informatik
Rapport nr. 2013:049
ISSN: 1651-476

Göteborgs universitet Institutionen för tillämpad informationsteknologi Göteborg, Sverige, Maj 2013

Abstrakt

Denna studie behandlar testarbete rörande utveckling och förändring av integrationsplattformar. Studien syftade till att undersöka vilka övergripande utmaningar som finns rörande testarbete vid utveckling och förändring av integrationsplattformar, samt vilka tankar som finns gentemot utvecklingsmetoden testdriven utveckling (TDD). En metod som betonar hög testtäckning och där testarbetet i hög nivå integreras med utvecklingsarbetet. För att söka svar på vår frågeställning genomfördes kvalitativa intervjuer tillsammans med företag och respondenter för att försöka förstå vilka övergripande utmaningar som finns relaterade till detta arbete och processer. De huvudsakliga slutsatser som ges i denna studie är att de övergripande utmaningarna finns i att involvera och utbilda projektdeltagare i testprocessen olika aktiviteter och vilka bidragande effekter test har på slutprodukten. En annan utmaning fanns i avsaknaden av en standardiserad testprocess som ses som ett krav för att uppnå ett strukturerat testarbete. Vårt resultat bekräftade dessutom de för- och nackdelar som beskrivs i tidigare litteratur rörande testdriven utveckling. **Nyckelord:** Integrationsplattformar, mjukvarutestning, riskbaserad testning, testdriven utveckling

Abstract

This thesis focuses on the testing activities at the stages of development and change regarding integration brokers. The study aimed to examine the overall challenges that exist regarding the testing activities within this context, and the thoughts towards the development method test-driven development (TDD). TDD emphasizes high level of test coverage and where the test process is well integrated with the development process. To seek answers to our question we applied qualitative interviews with companies and respondents to try to understand the global challenges that are related to this work and processes. The main conclusions presented in this study is that the overall challenges are involving and educating project participants in the testing process activities and the contributory effect test have on the end product. Another challenge was the lack of a standardized test process that is seen as a requirement to achieve a structured testing. Our results also confirmed the advantages and disadvantages as described in previous literature on test-driven development. **Keywords:** Integration brokers, software testing, risk-based testing, test-driven development

Tack

Vi vill tacka Enfo Zystems för hjälp att förverkliga denna studie. Tack också för hjälp med att skapa kontakter med företag som utmynnat i lärorika och intressanta samtal!

Vi vill också tacka alla respondenter som vänligt ställt upp och medverkat i vår studie.

Slutligen vill vi tacka vår handledare, Lisen Selander, för god och tydlig återkoppling och vägledning när det behövts.

Innehållsförteckning

1. Inledning

Vid en sammanslagning av företaget Norfolk Southern Corporation och Conrail förlorade Norfolk Southern Corporation mer än 113 miljoner dollar i intäkter, en stor del av denna förlust låg i problematiken kring mjukvarutestning. Den egenutvecklade logistikmjukvaran hade inte hade testats tillräckligt och dessutom hade felaktig testdata matats in i systemet. Detta resulterade i förlust av backup data, frakt som inte gick att spåra samt problem vid schemaläggning av personal. Företaget förlorade ytterligare 80 miljoner dollar på övertidstimmar för att åtgärda alla defekter innan systemet var stabilt igen, till följd av bristfällig test och verifiering av systemets funktion och beteende[1]. Norfolk-fallet är bara ett av många fall som påvisar svårigheterna med systemintegration och vikten av att testa systemets funktion och beteende mot existerande systemkrav.

I mitten av 2012 rapporterade CIO Sweden om en undersökning som Cordys låtit genomföra bland 650 europeiska beslutsfattare på såväl affärssidan som på ITsidan[2]. Målet för studien var att få en nulägesbild över europeiska beslutsfattares syn på hur väl deras IT stöttade affärsprocesserna. Det visade sig att 72 procent av beslutsfattarna på företagens affärsavdelningar inte ansåg att IT stödjer deras affärsprocesser i tillräcklig utsträckning. Skälen var både organisatoriska och tekniska och grundade sig i bland annat oflexibla och isolerade IT-system som inte stöttade processerna och samverkan mellan avdelningarna. För att undvika detta så menade man att bland annat att mer pengar och tid bör investeras på systemintegration. Integration och integrationsplattformar är en viktig teknologi för att sammankoppla system och verksamhetsprocesser och på så sätt uppnå effektivare flöden.

En fungerande system- och dataintegration mellan verksamheter och organisationers informationssystem blir alltmer viktigare (Overby et al, 2006). Föränderliga omvärldsvillkor ställer krav på organisationer och företag att kunna vara agila och ha möjlighet att snabbt kunna ställa om för att hantera förändringar (ibid). IT måste vara flexibelt för att kunna möta ständigt förändrade krav från verksamheten. Lösningen på dessa problem ligger ofta i en integrationslösning. Enligt Schmidt & Lyle (2010) kan integration definieras som: "Tekniken att

[1] http://www.computerworld.com/computerworld/records/images/pdf/44NfailChart.pdf

[2] http://www.idg.se/2.1085/1.453717/it-misslyckas-med-att-stodja-affarsprocesserna?queryText=systemintegration

sammanföra fristående system till en fungerande helhet.” Integrationsplattformar eller information brokers är system som möjliggör flexiblare data- och funktionsdelning mellan fristående informationssystem och skapar en grund för att göra information mer tillgänglig och användbar. På så sätt skapas en sammanhängande informationsinfrastruktur där tjänster och data blir mer tillgängliga vilket förbättrar effektiviteten och beslutsfattandet inom hela organisationen (Schmidt & Lyle, 2010).

Precis som andra programvaror har integrationslösningar utvecklings- och förändringsstadier under sin livscykel. Utveckling startar vanligtvis med att en beställning görs av den verksamhet som behöver en integrationsmöjlighet och en lösning tas fram tillsammans med leverantören. Efter lansering och implementation går lösningen över till förändringsstadiet där vanligtvis aktiviteter såsom underhåll och vidareutveckling utförs. Integrationsplattformar av olika slag kräver ofta olika typer av underhåll. Faktorer som påverkar kan exempelvis vara förändringar i IT-miljön, nya krav och behov av integrationskopplingar från affärsverksamheten eller mindre kod förändringar för att åtgärda defekter och problem (Overby et al, 2006).

En implementation av integrationsplattformar medför således både stora risker och kostnader. Kostnaderna innefattar främst olika former av mjukvaror och hårdvaror men också tid för både intern och eventuellt extern personal. Defekter och fel i produktionsmiljö kan medföra allvarliga konsekvenser för affärsverksamheten genom att orsaka driftstopp och kundklagomål. Tidigare forskning har funnit att defekter och fel kan vara så mycket som hundra gånger dyrare att åtgärda i produktionsmiljö än under utvecklingsstadiet. Det är således viktigt att defekter och fel identifieras och åtgärdas så tidigt som möjligt i utvecklingsprocessen (Iacob & Constantinescu, 2008). Agila utvecklingsmetoder såsom Testdriven utveckling (TDD) är utformad för att testa mjukvara pågående och redan vid utvecklingens start. TDD kan användas för att förhindra att defekter och fel sätter sig i koden från första början (Nikolik, 2012). Det är förekommande att integrationsprojekt inom IT blir försenade och överskrider sin budget. Konsekvensen av detta är att är att testaktiviteten som i sekventiella utvecklingsmetoder oftast ligger i de senare faserna i utvecklingsprocessen blir bristfällig (Amland, 2000).

1.1 Syfte och frågeställning

Vi vill med vår uppsats undersöka hur testarbete bedrivs rörande utvecklingen och förändring av integrationsplattformar. I detta ingår att undersöka vilka attityder samt svårigheter som finns relaterade till testarbetet. Vi vill dessutom ta reda på vilka tankar företag har kring testdriven utveckling vid utveckling av integrationsplattformar.

Utifrån denna bakgrund formulerar vi vår frågeställning:

"Vilka övergripande utmaningar finns rörande testarbete vid utvecklings- och förändringsarbete av integrationsplattformar?

1.2 Studiens upplägg

I teoriavsnittet integrationsplattformar och mjukvarutestning ges läsaren en introduktion till de teorier som uppsatsen baseras på. I metodavsnittet redogörs för vilka metoder och tillvägagångssätt som använts i uppsatsen för att komma fram till resultatet. I avsnittet presentation av empiriskt material presenteras resultatet från datainsamlingen. I analysavsnittet kommer relationen mellan teori och empiri jämföras och tolkas var på sedan slutsatser presenteras under slutsatsavsnittet.

2. Integrationsplattformar och mjukvarutestning

I teoriavsnittet kommer vi att redovisa och diskutera tidigare forskning och annan teoribildning för att beskriva och förklara centrala begrepp inom test av integrationsplattformar. Dels för att ge oss själva ökad insikt och förståelse för problemområdet, dels för att beskriva begrepp som är viktiga för läsaren att förstå. Det teoretiska ramverket grundar sig i omfattande litteratursökningar i relevanta artikeldatabaser. Litteratursökningar har inledningsvis placerats i ämnesområdet för utveckling och förändring av integrationsplattformar samt riskhanterings- och testprocessen kring dessa.

2.1 Integrationsplattformar

Inom organisationer återfinns ofta flera olika system som integrerar med varandra. Integrationsplattformar eller så kallade integration brokers fungerar som ett nav och har till

uppgift att förmedla information mellan de olika systemen. Med tanke på att all informationsutväxling mellan systemen går via integrationsplattformen blir det enklare då ett nytt system skall implementeras inom organisationen, då bara en koppling måste skapas mellan det nya systemet och plattformen. Mule ESB, Microsoft BizTalk, och IBM Websphere är olika fabrikat på integrationsplattformar som har som huvudfunktion att översätta utgående data från ett system och behandla det utifrån en uppsättning regler, till exempel att konvertera en fil till annat filformat.

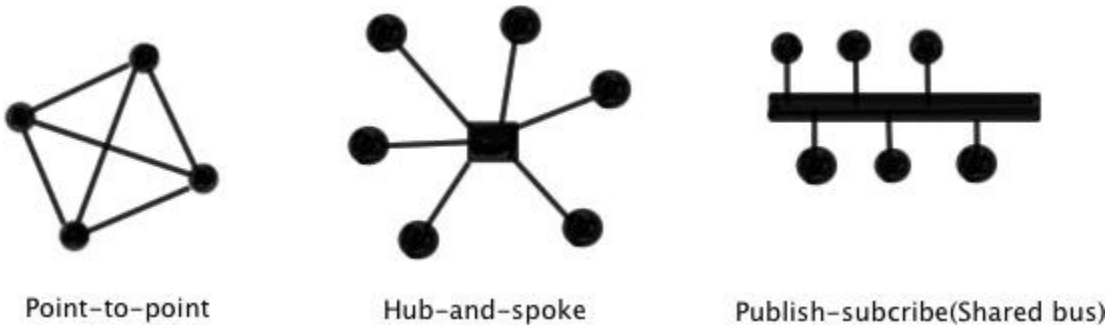
Flera tekniker och ramverk har genom åren utvecklats för att möjliggöra integration. En central teknik inom detta segment är Enterprise Application Integration (EAI). Syftet med EAI är att motverka den isolering av data och funktioner, som är typisk vid olikartade applikationer. EAI är inte en enskild teknologi utan baseras på en mängd olika verktyg och tekniker såsom message broker och XML, för att möjliggöra process och datadelning mellan applikationer. Valet av teknik eller verktyg bör återspeglas i den aktuella IT-arkitekturen och dess strategi (Papazoglou & Ribbers, 2006). EAI definieras som "the combination of processes, software, standards, and hardware resulting in the seamless integration of two or more enterprise systems allowing them to operate as one" (Fenner, 2013 s1).

Beroende på hur organisationens nuvarande IT-arkitektur är uppbyggd bestäms lämplig EAI-topologi, dessa topologier är point-to-point-, hub and spoke- och publish subscribe (Shared bus)-topologi (Linthicum, 1999).

- **Point-to-point:** Är integration i sin enklaste form. Kopplingspunkter sätts upp mellan de noder som skall kommunicera med varandra. Denna topologi är mest lämpad vid integration av några få system på grund av den exponentiella ökningen av kopplingspunkter.

- **Hub and spoke:** Här sker integreringen med hjälp av en central integration broker(hub) som kopplar ihop noderna. Brokern översätter datan så att den kan behandlas och hänvisar till applikationen som eftersöker den.

- **Publish subscribe(Shared bus):** Här sker integrationen med hjälp av en gemensam nätverksbuss och möjliggör på så sätt delad kommunikation (Linthicum, 1999).



Figur 1. Illustration av Enterprise application integration topoligierna.

EAI består av fyra olika typer av integrationslösningar.

- **Data-level:** Är den process för att flytta data mellan olika datalager och på så sätt göra data åtkomligt för flera fristående system.
- **Application interface-level:** Är den process där man använder sig av gemensamt gränssnitt som de fristående applikationerna utnyttjar för att dela funktion och data, detta är en lösning hos affärssystem som ofta består av integrerade moduler.
- **Method-level:** Denna teknik används för att dela affärslogiken mellan applikationer, detta möjliggör återanvändning av metoder vilket reducerar produktionstiden.
- **User interface-level:** I denna process skapar man ett nytt gränssnitt som man sedan använder för att koppla samman de olika systemen (ibid).

EAI är en pågående process under hela förändringsprocessen för att möjliggöra implementering av nya applikationer i den befintliga IT-infrastrukturen (Papazoglou & Ribbers, 2006).

2.2 Riskbaserad testning

Utvecklingsprojekt arbetar ofta under begränsade resurser i form av tid, budget och människor (Alam & Khan, 2013). Utvecklingsprojekt är sällan inom ramarna för tid och budget. Ofta på grund av försening i tidigare projektfaser. En konsekvens av detta är att när det så småningom kommer till testning, är tiden tills leveransen extremt kort och det inte finns budget kvar till att tillräcklig testa alla funktioner (Amland, 2000). I riskbaserad testning utgår man från de risker

som finns relaterade till produkten och dess kontext, i syftet att optimera teststrategin (SWEBOK:2005). Detta stöds av Amland (2000) som menar att idén om riskbaserad testning är att fokusera på testaktiviteter och spendera mer tid på att testa kritiska funktioner. En risk kan beskrivas som en osäkerhet kring ett möjligt framtida utfall, som anses negativt. Om en risk kommer förekomma eller inte kan aldrig garanteras på förhand, men det kan neutraliseras genom förebyggande åtgärder (Alam & Khan, 2013).

Metoden för riskbaserad testning beskrivs i ett antal sekventiella steg. I det första steget beskriver man vilka risker som finns och hur de kan uppstå associerade till de ställda kraven. I andra steget rangordnas kraven utifrån dess riskprofil. I det tredje steget upprättar man en handlingsplan för att hantera de beskrivna riskerna. I det fjärde steget verkställs handlingsplanen och testaktiviteterna startar och följs sedan upp (ibid).

2.3 Mjukvaruvarutestning

Mjukvarutestning är en pågående aktivitet som utförs längs utvecklings- och förändringsprocessen för att utvärdera en programvaras kvalitet, och för att förbättra den, genom att identifiera defekter och problem. Enligt Iacob & Constantinescu (2008) är test och kvalitetssäkring bland den högst prioriterade frågan hos chefer för mjukvaruutveckling. Inom en systemutvecklingskontext kan kvalitet beskrivas som hur väl produkten uppfyller eller överensstämmer med de funktionella kraven (vad systemet ska göra). Dessutom till vilken grad programvaran uppfyller de ickefunktionella kraven (t.ex. prestandakrav eller säkerhetskrav). Defekter definieras som någonting som orsaker ett fel under exekveringen av ett program, vilket kan leda till misslyckande om de inte identifieras och åtgärdas före lansering (SWEBOK:2005). Defekter uppstår eftersom människor gör misstag och även på grund av tidspress, kod komplexitet, förändrade tekniker och miljöförhållanden (Iacob & Constantinescu, 2008). Mjukvarutestning kan särskilt kategoriseras inom statiska och dynamiska tester. I statiska tester sker ingen kodexekvering utan programvaran utvärderas med användning av tekniker såsom recensioner, statisk kodanalys och inspektioner. I dynamiska tester som ligger i fokus för denna uppsats, sker kompilering och exekvering av programvarukoden med en given uppsättning testfall för att utvärdera dess dynamiska beteende. Testfall är ett mångsidigt verktyg som används för att hantera mjukvarudefekter genom att förebygga fel, visa förekomst och avsaknad av defekter, isolera eller lokalisera defekter och undvika defekter (Nikolik, 2012). Ett testfall

består vanligtvis av en uppsättning villkor eller variabler under vilka en testare avgör om en mjukvaras beteende är korrekt och efter förväntan, genom att jämföra faktiskt utdata mot ett förväntat resultat. Ett testfall med giltiga variabler benämns ofta som ett “good case” och testfall med ogiltiga variabler som ett “bad case” (ibid).

Mjukvarutestning utförs vanligtvis på olika nivåer längs systemutvecklings- och förändringsprocessen. Anledningen till detta är att testobjektet, det vill säga vad som ska testas, kan variera från en enstaka enhet (klass eller metod) till hela system. Syftet med testet kan också variera beroende på vilken nivå testobjektet befinner sig på. Regressionstester kan exempelvis genomföras under alla nivåer och syftar till att verifiera att modifikationen av en del mjukvara inte har orsakat några oavsiktliga effekter. Funktionstester kan genomföras på högre nivåer för att verifiera att de funktionella specifikationerna genomförs korrekt. Lämpligheten hos tester och testfall varierar dessutom med typen av mjukvara. För en generell datorspelsprodukt är troligtvis alfa och beta testning viktigt under utvecklingsprocessens senare faser för att få hjälp av användarmassan med att hitta problem och defekter. Men för en specifik integrationslösning för en unik organisation är det möjligtvis mer lämpligt att tilldela resurser för acceptanstestning av systemet efter leverans (SWEBOK:2005).

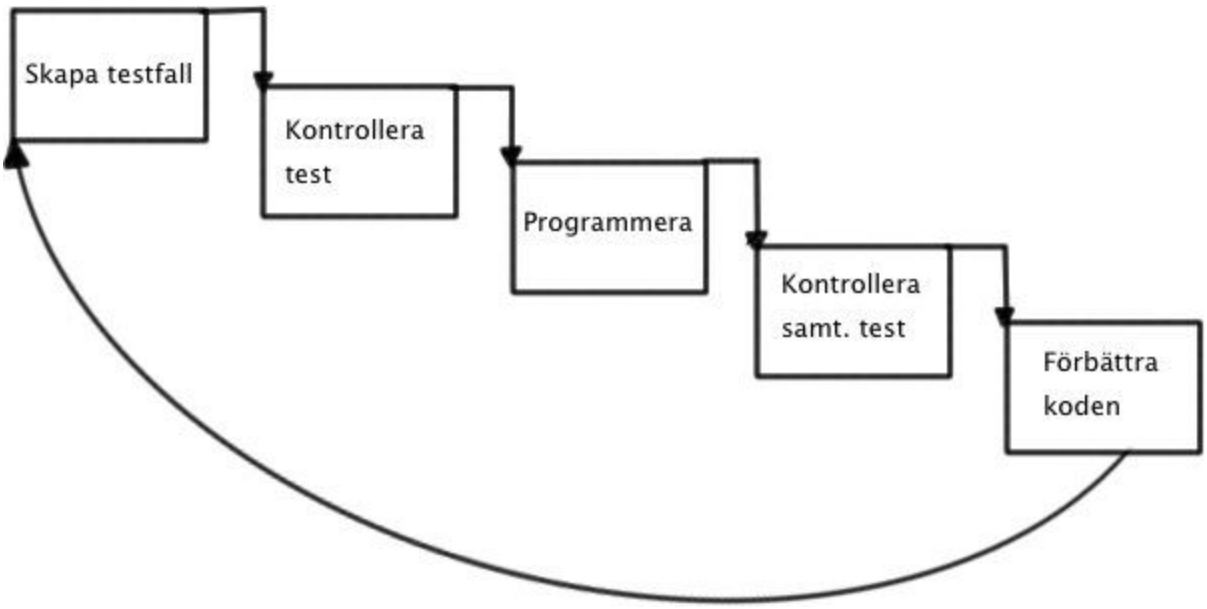
Traditionellt indelade testnivåer inom dynamisk testning enligt standard dokumentationen för systemutveckling (SWEBOK:2005).

- **Enhetstester:** Verifierar funktionen av en isolerad del mjukvara som testas separat, så kallade enhet. Grupper av tätt relaterade enheter kan utgöra enskilda komponenter. Enhetstestning sker direkt i koden av utvecklaren, ofta med hjälp av testramverk för enhetstestning (Larman, 2004).
- **Integrationstester:** Sker efter enhetstestning och är processen för att verifiera att interaktionen mellan integrerade komponenter, bestående av relaterade enheter, fungerar som förväntat.
- **Systemtester:** Avser beteendet hos ett helt integrerat system för att kontrollera det mot användarnas förväntningar och mot specifikationen eller kraven.

Testaktiviteterna måste integreras i en definierad och kontrollerad process som drivs av människor (SWEBOK:2005). Enligt Iacob & Constantinescu (2008) består den traditionella testprocessen av huvudaktiviteterna *planering och kontroll, analys och design, implementation och utförande* och *utvärdering av slutkritetrier för testning och rapportering*. Detta stöds av Khan & Choudhary (2011) som beskriver ett liknade sekventiellt ramverk för testprocessens huvudaktiviteter. I kontrast till de traditionella testprocesserna är de agila såsom testdriven utveckling formad för att testa mjukvara pågående och redan vid utvecklingens start. Det innebär att kvalitetsansvaret förflyttas från att till största delen ligga hos testarna och tillämpas i utvecklingsprocessen olika faser, till att ligga mer hos utvecklarna (Ganesh & Thangasamy, 2012).

2.4 Testdriven utveckling

Test-driven development (TDD) eller testdriven utveckling är en systemutvecklingsmetod (Beck, 2003) där utvecklaren först skapar enhetstester, för att sedan skriva kod som får dessa tester att passera. En viktig princip är att varje testfall ska testa en individuell aspekt av en klass (enhet) istället för att testa helheten. Metoden är iterativ och bygger på mindre cyklar. När en utvecklare skall implementera en ny funktion börjar han med att skriva ett testfall. För att utvecklaren skall kunna konstruera ett test fordrar det att denne är helt införstådd med funktionens utförande och krav. Från början kommer testet misslyckas då ingen kod har skrivits, om det inte misslyckas finns antingen funktionen redan i applikationen eller så är testet felkonstruerat. Utvecklaren kommer nu att programmera funktionen så att den skall klara testet han skapat. När testet väl går igenom kontrollerar utvecklaren samtliga tidigare testfall för att se att tidigare funktioner är intakta och inte påverkats av den nya funktionen, denna process kallas för regressionstestning. Utvecklaren kan nu vid behov förbättra koden utan att förändra dess grundläggande funktion, denna process kallas refaktorering. Målet med refaktorering är att ha precis så mycket kod som krävs för att funktionen framgångsrik skall kunna exekveras (Larman, 2004). När cykel är genomförd startas en ny från början med samma steg.



Figur 2. Överblick av den testdrivna utvecklingsprocessen.

En grundregel inom TDD är att om du inte kan skriva ett testfall för vad du tänker koda, så borde du inte ens tänka på att koda (Chaplin 2001 se Williams & George 2004). Tidigare studier av Williams & George (2004); Erdogmus et al (2005); Larman (2004) och Nagappan et al (2008) har påvisat ett antal för- och nackdelar som uppstår vid användandet av TDD. Fördelar finns i bland annat ökad effektivitet, möjligheten att lagra och dokumentera test tillgångar såsom testfall och testskripts samt reducering av kod defekter.

> **Effektivitet:** De korta och mindre cyklar som kännetecknar testdriven utveckling möjliggör kontinuerlig feedback till utvecklarna vilket skapar en högre grad av kontroll och verifikation av kodens beteende och funktion. Defekter och fel kan identifieras tidigt och snabbat allteftersom ny kod adderas till systemet och utvecklarna kan snabbt åtgärda eventuella problem som uppstår. Enligt Larman (2004) kan de enhetstester som skrivs för alla produktionsklasser som skall testas automatiseras och köras efter varje kodförändring, i ett så kallat regressionstest. Genom att kontinuerligt köra dessa automatiserade enhetstester, kan man få tidig feedback på om en förändring bryter befintlig funktionalitet, snarare än att upptäcka detta i utvecklingens senare skeden. Följaktligen är källan till defekten lättare att fastställa och mindre tid behöver läggas på felsökning, eftersom utvecklaren har kännedom om var och när felet uppstod. Williams & George (2004)

hävdar att TDD's effektivitet i att förhindra och eliminera defekter, minskade resurs- och tidsåtgång vid felsökning och förändringsarbete, kompenserar för den extra tid som går åt att skriva och exekvera testfall och utföra enhetstester.

Test tillgångar: Det faktum att testfall för enheter alltid skrivs före själva koden, försäkrar att enhetstester faktiskt skrivs vilket är skapar värdefulla tillgångar för projektet (Williams & George, 2004). Enligt Larman (2004) kommer det tillslut finns hundratals eller tusentals enhetstester, vilka kan lagras i en så kallad testsvit och återanvändas under utvecklings- och förändringsprocessen för att verifiera att kodförändringar inte orsakar fel eller defekter i systemet.

Nackdelar finns i bland annat övertro till metoden och svår tillämpning av arbetssättet.

Övertro till metoden: Det stora antalet passerande enhetstester kan ge en falsk känsla av trygghet, vilket resulterar i färre och varierande testaktiviteter. TDD vilar på enhetstester som sker på lågnivå och på mindre portion programkod. Även om koden passerar alla tester på denna nivå är det inte självklart att integrationen mellan relaterade komponenter (bestående av flera relaterade enheter) fungerar. Därför är också integrations- och funktionstester viktiga aktiviteter (Nagappan et al, 2008).

Svår tillämpning: TDD har en lång inlärningskurva. Tidigare studier studier har visat att utvecklare ansåg TDD vara en av de svåraste utvecklingsmetoderna att implementera, och att det kräver en viss inställning till arbetet. Mycket tid behövdes i början för att skriva testfall före produktionskod (Dogsa & Batic, 2011).

Mycket av den forskning som skett på området fokuserar på kontrollerade experiment för att undersöka effekten av att arbeta utifrån en testdriven metod kontra mer traditionella utvecklingsmetoder, där testning och kvalitetssäkring sker i de senare faserna av utvecklingsprocessen (Ganesh & Thangasamy, 2012). I en studie av Dogsa & Batic (2011) observerades tre projekt, varav två arbetade utan TDD och det tredje introducerades för det. Resultatet indikerade att projektet som arbetade testdrivet producerade högre kod kvalitet som är lättare att underhålla, även om de såg en minskning av produktiviteten. I en annan studie av

Williams & George (2004) genomfördes ett strukturerat experiment av TDD. I studien utvecklade en grupp ett mindre Javaprogram med hjälp TDD medan den andra (kontrollgruppen), använde ett vattenfallsliknande tillvägagångssätt. Även här visade det sig att utvecklare som arbetat efter det testdrivna arbetssättet producerade högre kod kvalitet. Dock tog de 16 % mer tid på sig att utveckla applikationen. Annan tidigare forskning pekar på att kostnaden för att åtgärda buggar och fel under utvecklingsprocessen är avsevärt mycket lägre än efter konstruktionen eller till och med efter lanseringen (Nikolik, 2012; Iacob & Constantinescu, 2008). I studie av Nagappan et al (2008) antog tre utvecklingsgrupper TDD. Studiens resultat indikerade att defekt intensiteten före lansering minskade mellan 40 % och 90 %, i förhållande till liknande projekt som inte använder TDD i praktiken. Även i denna studie upplevde grupperna en 15-35 % ökning av utvecklingstiden efter antagandet av TDD.

3. Metod

I metodavsnittet kommer vi att redogöra för den metod som användes för att söka svar på vår frågeställning: *"Vilka övergripande utmaningar finns rörande testarbete vid utvecklings- och förändringsarbete av integrationsplattformar?* För att svara på våra frågeställning valde vi att använda oss av en kvalitativ undersökningsmetod (Bernard, 2011), där vårt främsta verktyg för att samla in data var intervjuer. Intervjuerna ägde rum tillsammans med beslutsfattare från företag som arbetar med eller håller på att bygga upp ett mer robust testarbete. Examensarbetet skrevs i samarbete med IT-konsultbolaget Enfo Zystems som introducerade oss för problemområdet. Genom Enfo Zystems breda nätverk och kunddatabas fick vi tillgång till de företag och respondenter som medverkat i den empiriska studien.

3.1 Vetenskapligt förhållningssätt

Enligt Patel & Davidsson (2011) är positivismen och hermeneutistiken två aktuella vetenskapliga förhållningssätt. Positivismens idéer härrör från en empirisk naturvetenskaplig tradition. Karaktäristika hos positivismen är idén om reduktionism, alltså att helheten av ett problem kan brytas ner i dess utgörande beståndsdelar och studeras enskilt för att sedan skapa en förståelse av helheten (ibid). Grundläggande uppfattningar inom positivismen är att kunskapen om det faktiska förhållandet skall baseras på empiri och att tidigare erfarenheter eller kunskap hos

subjektet inte skall kunna påverka forskningsresultatet. Uppfattningarna återfinns i vetenskaper som fysik och matematik (Nationalencyklopedin, 2013).

I kontrast till positivism har hermeneutiken ett mer holistiskt synssätt där helheten anses vara större än summan av delarna, då samspelet mellan delarna skapar en större helhet. En hermeneutiker försöker först förstå helheten för att sen titta närmare på utgörande delar och skapa en förståelse kring dessa. Dennes tidigare kunskap, känslor samt personliga värderingar är ett centralt redskap vid tolkningsprocessen (Patel & Davidsson, 2011). Forskning inom området informationssystem, där kunskap anskaffas genom sociala konstruktioner, klassificeras som en tolkande forskning (Klein & Mayers, 1999). Vi lät oss inspireras av det hermeneutiska synssättet och valde en tolkande ansatts för arbetet. Då vi ansåg att det positivistiska synsättet lämpade sig bättre vid naturvetenskapliga forskningsområden där verifierbarhetsprincipen råder, alltså där allt ska kunna översättas till verifierbara observationer. Vår frågeställning var sådan att vi ansåg den bli påverkad av mänskliga faktorer som inte är mätbara i sin natur, såsom tankar, känslor och värderingar.

3.2 Litteraturstudien

Vår litteraturstudie var iterativ och pågående i stort sätt genom hela studien. Detta eftersom kunskapen och förståelsen för problem- och undersökningsområdet hela tiden ökar allteftersom mer information inhämtas (Patel & Davidsson, 2011). Problem- och undersökningsområdet vilken studien rör sig inom bestämdes i samverkan med Enfo Zystems, som examensarbetet skrevs i samarbete med. Tillsammans med Enfo Zystems identifierades viktiga begrepp såsom integration, integrationsplattformar, mjukvarutestning, test-driven utveckling. Vi studerade vetenskapliga artiklar och böcker, vars innehåll överensstämde med de viktiga begreppen. Litteratursökningarna resulterade i ett 15-tal artiklar och böcker. Materialet studerades överskådligt för att effektivt kunna välja bort material som vi ansåg vara irrelevant. Litteraturförteckningarna i dessa artiklar och böcker gav också tips på annan litteratur. Som sökverktyg använde vi oss främst av databasen SUMMON som tillhandahålls av Göteborgs Universitetsbibliotek och Chalmers bibliotek. Vi har också använt oss av tidigare kurslitteratur från det systemvetenskapliga programmet samt genomfört sökningar i Google Scholar efter rekommenderade artiklar som inte hittats i SUMMON. Litteraturstudien hade en stor betydelse

för undersökningen och gav oss ökad förståelse för problemområdet och vilka teorier och begrepp som är centrala. Detta utgjorde grunden för vår teoretiska referensram.

3.3 Datainsamlingsmetod

För att söka svar på vår frågeställning valde vi att använda oss av en kvalitativ undersökningsmetod (Bernard, 2006), där vårt främsta verktyg för att samla in data var semistrukturerade intervjuer. Semistrukturerade intervjuer möjliggör framträdandet av detaljkunskap och en djupare förståelse för det problemområde som studien rör sig inom, genom att tillvägagångsättet grundar sig på öppna frågor som rör sig inom begrepp, teman och teorier som anses vara viktiga för studiens syfte och frågeställning (Patel & Davidsson, 2011). I kontrast till strukturerade intervjuer där förbestämda frågor och svar givits, tillåter den semistrukturerade intervjun följdfrågor på de svar som respondenten ger (Bernard, 2006). Detta gav oss en större förståelse för respondentens svar och på så sätt uppstod goda förutsättningar att komma ner till kärnan av de bakomliggande drivkrafterna. De semistrukturerade intervjuerna genomfördes antingen på plats eller över telefon tillsammans med respektive respondent. För samtliga intervjuer tog vi hjälp av en intervjuguide (se bilaga 1) för att inte glömma viktiga aspekter och frågor. Intervjuguiden baserade vi på dels vår insamlade teori, dels våra antaganden om det givna problemområdet. Intervjuerna inleddes av en formell presentation av alla intervjudeltagare och en bakgrundshistoria av respondenten och dennes roll i organisationen. Samtliga intervjuer spelades in med respondentens godkännande. Anledningen till inspelningen var att fokus skulle ligga på själva samtalet och inte störas paus för anteckningar. Respondenterna hade endast informerats om temat och tidsåtgången för intervjun, men inte tagit del av intervjuguiden.

3.4 Om Enfo Zystems

Enfo Zystems är ett IT-konsultbolag och specialister när det gäller systemintegration av applikationer, affärsprocesser och informationssystem. Företaget erbjuder paketerade fullservice-lösningar och utgår från ”Best practice” i integrationsprocessen- genom hela livscykeln. Företaget har som affärsmodell sälja och administrera integrationslösning och plattformar. Det betyder att företaget innehar stor kunskap inom området som anskaffats genom otaliga integrationsprojekt.

- Enfo Zystems erbjuder Europas största Center of Excellence för support och förändring av integrationsplattformar byggda på IBM, Microsoft eller MuleSoft integrationsprodukter
- Flexibla och underhållsvänliga integrationslösningar, vare sig det handlar om A2A (application to application), B2B (business to business) eller Molnet. Baseline metodiken används vid integrationsprojekten tillsammans med företagets egna arbetssätt.
- Inom området Business Process Management (BPM) hjälper de kunder att effektivisera och skapa förbättringsbara och mätbara affärsprocesser.
- Enfo Zystems IDC (Integration Delivery Center) erbjuder integration som en managerad tjänst. Kunder kan outsourca sitt integrationsarbete till Enfo, helt eller delvis.

3.5 Urval

Eftersom Enfo Zystems specialiserar sig inom integrationslösningar, valde vi att använda oss av företagets kontaktnätverk och kunddatabas för att hitta lämpliga företag för undersökningen. Tillsammans med vår kontaktperson på Enfo Zystems och ett antal urvalskriterier tog vi fram möjliga kandidater. Undersökningen rörde sig inom problemområdet för test av integrationsplattformar. Detta krävde att företagen utövade någon form av testarbete rörande utveckling och förändring av integrationsplattformar. Inom dessa företag riktade vi oss mot personer som är insatta i, och har erfarenhet av, utveckling och förändring av integrationsplattformar. Urvalsprocessen utmynnade i fyra företag. Vår kontaktperson ansvarade sedan för att etablera kontakt och samordna intervjutillfällen, varav vi sedan genomförde dessa intervjuer antingen på plats eller över telefon.

Nedan följer en kort beskrivning av samtliga företag som svarade mot de uppsatta kriterierna, vilka hålls anonyma samt könsneutrala i vår beskrivning.

Företag 1: Ett IT-konsultbolag som specialiserar sig inom systemintegration av applikationer, affärsprocesser och informationssystem. Respondenten har lång erfarenhet inom integrationsprojekt och stor insikt kring testprocessen av dessa.

Företag 2: Ett företag inom bankväsendet. Företaget håller för nuvarande på att bygga en robust infrastruktur för strukturerad testning. Företaget har en generell testprocess som

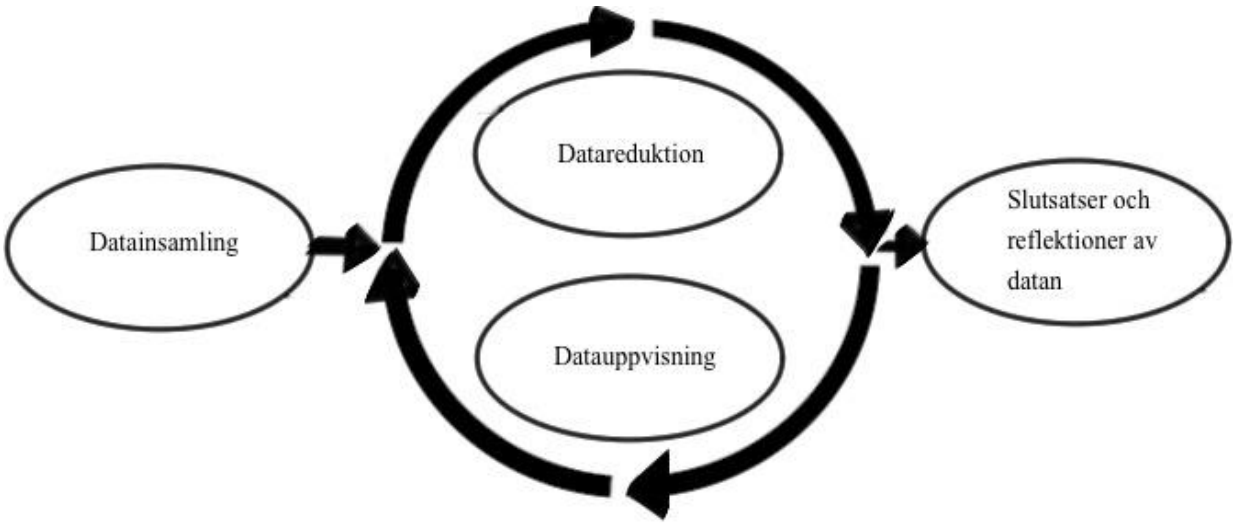
gäller för alla typer av utvecklingsprojekt. Respondenten är ansvarig för bankens testprocess.

Företag 3: Ett företag inom tillverkningsindustrin. Företaget har en generell testprocess och en gemensam värdegrund för hur test av mjukvara skall bedrivas. Respondenten har haft många roller och jobbat med integrationsutveckling som mjukvaruarkitekt i många år. För tillfället arbetar respondenten med integrationsutveckling på olika sätt och är insatt i testarbetets olika delar.

Företag 4: Ett företag inom IT/Telekom. Företaget arbetar aktivt med efter utsatta standarder vid test av mjukvara och integrationsplattformar. Respondenten har arbetat med olika roller inom testning av integrationsplattformar i cirka åtta års tid. Arbetat bland annat som integrationsutvecklare med fokus på test. Dessutom utvecklat test-strategier och rutiner för hur man ska arbeta med test såsom testmetodik och testverktyg.

3.6 Hantering av empiriskt material

Våra semistrukturerade intervjuer resulterade i en stor mängd ostrukturerad data. Huberman & Miles (1994) föreslår en metodik för kvalitativ dataanalys bestående av tre delar; datareduktion, datauppvisning samt slutsatser och verifikation.



Figur 3. Illustration av den kvalitativa dataanalysmetodiken (Huberman & Miles, 1994).

Datareducering refererar till processen av att välja, förenkla, abstrahera och transformera den data som genererats från datainsamlingen. Denna process förekommer kontinuerligt under studiens livscykel, även före och efter datainsamlingen. Detta beror på att forskaren både medvetet och omedvetet hela tiden väljer vilken information som skall följas upp och på det sättet sker en datareduktion (Huberman & Miles, 1994).

De intervjuer som hölls spelades in för att sedan transkriberas och bearbetas. Datan reducerades genom att vi tog ut centrala faktorer som vi ansåg gav oss svar på vilka övergripande tankar företagen hade kring testarbete rörande utveckling och förändring av integrationsplattformar. Dessutom frågor som rörde hur testarbetet bedrevs och tankarna kring utvecklingsmetoden testdriven utveckling.

Datauppvisning refererar till hur forskaren väljer att presentera sin komprimerad och organiserad information för att lättare kunna utläsa mönster och dra slutsatser. Vår datareduktion resulterade i textstycken där vi sammanställde vår tolkning av respondentens tankar och förhållning kring vad vi ansåg vara centrala faktorer inom problemområdet. Detta gav oss en förutsättning för att kunna förstå materialet och kunna bestämma om materialet var moget nog för att basera våra slutsatser på eller var i behov av att analyseras ytterligare.

Slutsatser och reflektioner är den tredje delen i (Huberman & Miles, 1994) kvalitativa dataanalysmetod där slutsatser och tillhörande reflektioner görs på det behandlade materialet, vilket vi i vår uppsats redogör för i resultatredovisning och analys.

4. Presentation av empiriskt material

I avsnittet redovisas resultaten av vår genomförda undersökning. Företagen redovisas sekventiellt.

4.1 Företag 1

Företag 1 är ett IT-konsultbolag som främst arbetar med systemintegrationer av applikationer, affärsprocesser och informationssystem. Vår respondent har en lång erfarenhet inom integrationsprojekt och besitter en stor förståelse kring testarbete.

Idag arbetar de med test vid integrationsutveckling genom att utvecklaren får ett mer eller mindre väldefinierat uppdrag att till exempel koppla ihop ett antal system. I bästa fall får utvecklaren ett exempel på ett frisk meddelande in och exakt hur det skall vara formaterat när det kommer ut. I sämsta fall får utvecklaren endast ett textdokument som beskriver filen in och då måste utvecklaren snickra ihop en testfil, innehållande testdata på egen hand för att överhuvudtaget kunna genomföra testaktiviteterna.

”98 procent av alla integrationslösningar testas genom att man kör igenom en eller två good cases för att se att det fungerar. Man måste istället fråga sig vad som kan gå fel”.
De ser en överhängande attityd hos många av deras beställare, att ogärna betala för testarbete och därmed kvalitetssäkring. Attityden grundar sig ofta i okunskap om testinsatsens betydelse och låg riskmedvetenhet. Samtidigt inser de att för att kunna säkerställa att en robust testning genomförs, krävs det inte bara en teknisk infrastruktur som stödjer detta, utan också standardiserade arbetssätt som behandlar processen kring testning. Standardiseringen eller regelverket skulle definiera strategiska mål med testinsatsen; vad som skall testas, vem som ansvarar samt vem som följer upp de utsatta nyckeltalen.

”Testfallen kan återanvändas vid framtida förändringar”.
Företaget känner till och ser ett antal fördelar med test-driven utveckling eftersom det betonar hög testtäckning och kontroll över den kod som skrivs. Fördelarna de ser är att testfall och enhetstester för alla produktionsklasser faktiskt skrivs av utvecklarna, och att testfallen sedan kan återanvändas både under utvecklingsprocessen och vid framtida förändringar vid underhållsarbetet. Under utvecklingsprocessen ser företaget möjligheten att bygga upp en testfallsbank av enhetstester. Enhetstesterna kan sedan köras automatiskt efter varje refaktorering av en implementerad klass för att verifiera att dess funktion och beteende är intakt. Dessutom kan testerna köras för att verifiera att andra relaterade klasser inte blivit negativt påverkade av förändringen. Testbanken hade också kunnat återanvändas vid större förändringar där man måste kunna säkerställa att funktioner och kopplingar både fungerat tidigare men också fungerar efter förändringen.

4.2 Företag 2

Företag 2 verkar inom bankväsendet och håller för tillfället på att bygga en robust infrastruktur för strukturerad testning. De har en generell testprocess som gäller för alla typer av utvecklingsprojekt inom organisationen. Vår respondent är ansvarig för framtagningen standardiseringen av organisationens testprocess.

"Ett av företagets högst prioriterade områden är leveransen och dess kvalitet samt stabilitet".

Företaget har en intern stöd- och supportorganisation med ansvarsområden som innefattar kravprocess, testprocess, releaseprocess, testverktyg samt test av funktionella och icke-funktionella krav. Sedan 2006 har företaget prioriterat och sett över sin testprocess för att förbättra den. Ett av företagets högst prioriterade områden är leveransen och dess kvalitet samt stabilitet. Testarbetet är inte begränsat till endast utveckling utan täcker dessutom förvaltning, infrastruktur och programvara från tredjepartsleverantörer. De har infört ett standardiserat arbetssätt som gäller vid alla projekt och förändringar. För varje testnivå finns det ett antal slutkriterier som minst ska vara uppfyllda. Sedan finns det också flera checkpunkter längs utvecklingsprocessens väg för att säkerställa att man har kännedom om projektets status. Respondenten menar att det är viktigt att ha tre saker klara för sig; Vad leveransbeskrivningen innehåller, vad målet är och hur du ska nå dit. Man ska kunna kontinuerligt kontrollera var i processen man är, vilka mål man har uppnått. Hur väl detta kan kontrolleras faller tillbaka på hur bra de initiala kraven var och om det finns brister i kraven eller om kraven är allt för rörliga. För att öka chansen till att upptäcka eventuella brister och missar i kraven genomför företaget gemensamma leveranstester. Under de gemensamma leveranstesterna körs regressionstester samt tester av alla möjliga typer av icke-funktionella krav, såsom prestandatester. I dessa sammanhang kan projekten också efterfråga olika typer av tester som de anser vara viktiga att genomföra före leverans. Tester genomförs vilket man anser är bra, men företaget testar systemet oftast ur ett positivt perspektiv, man verifierar att funktionaliteten fungerar i ett normalfall. De anser att mer tid och fokus borde ligga på vad de kallar negativa tester för att exponera brister och svagheter i mjukvaran.

"Den testdrivna utvecklingsmetoden är ingen helig gral"

Företaget eftersträvar ett arbetssätt likt testdriven utveckling. Respondenten menar att en sådan förändring tar tid eftersom organisationen är stor samt att flera svårigheter måste hanteras. Svårigheter ligger i att utvecklarnas attityd gentemot den testdrivna utvecklingsmetoden måste förändras för att metoden skall kunna användas fullt ut. Endast en metod är emellertid inte lösningen på alla problem. Testdriven utveckling betonar hög testtäckning men kan skapa en falsk känsla av trygghet. Respondenten menar att en variation av testtekniker, positiva och negativa testfall samt test på flera testnivåer krävs för att åstadkomma god testtäckning.

"Man måste göra kloka estimeringar och avsätta tid till testaktivieter".
Respondenten menar att det går en negativ trend i att tid avsatt för testaktiviteter ofta försvinner när utvecklingen blir försenad. Testledaren avsätter ofta tid för testaktiviteter. När utvecklingen blir försenad tas tid, avsatt för testaktiviteterna. Enligt företaget måste medarbetarna lära sig att estimera bättre, om inte får de kapa i kravspecifikation för att hinna med test av viktiga funktioner.

4.3 Företag 3

Företag 3 är ett företag inom tillverkningsindustrin. Företaget har en generell testprocess och en gemensam värdegrund för hur test av mjukvara skall bedrivas. Vår respondent har lång erfarenhet och haft flera roller inom integrationsutveckling. För tillfället arbetar respondenten med integrationsutveckling på olika sätt och är insatt i testarbetets olika delar.

Företaget har en intern funktion som kallas för integrations center bestående av 180 personer som arbetar inom olika roller, såsom arkitekter och utvecklare som är specialiserade inom integrationslösningar. Organisationen har även en styrande funktion som tar IT-strategiska beslut och arbetar med framtagning av metoder rörande integrationsutveckling. Det kan handla om till exempel, vilka arkitektuella principer och utvecklingsmetoder som skall eftersträvas. I dagsläget genomför de inte enhetstester som är testning av enskilda enheter (klasser och metoder) på grund av att deras testverktyg inte tillåter detta. Idag sker testningen på en högre nivå där komponenter som utgörs av relaterade enheter verifieras i integrationstester. Detta gör man genom att använda ett antal good cases och bad cases för säkerställa att interaktionen mellan komponenterna fungerar. De anser att fullständig testning inte är genomförbart. Anledningen är att man hela tiden arbetar inom begränsningen för tid och resurser, under dessa förutsättningar måste en

medvetenhet finnas för vad som är tillräckligt hög testteckning. Företaget har arbetat fram riktlinjer som de kallar happyflow som adresserar vikten av att välja ut och testa huvudfunktioner. Graden av testning bestäms utifrån mjukvarans ändamål, baserat på hur avancerad logiken är. När happyflow inte är tillräckligt skrivs fler testfall för att uppnå högre testteckning.

"Användare litar ofta på att programvaran är korrekt och tror att fel som uppstår har orsakats av dennes handlingar"

Efter deras egna erfarenheter kan de konstatera att kostnaderna för korrigering av defekter och fel ökar markant efter lanseringen av två anledningar: Antalet mantimmar det tar från det att en defekt uppdagas av slutanvändaren tills det rapporterats. Användaren ofta litar på att programvaran de använder är korrekt och ser då till sig själv och antar att de orsakat felet. Det anses också vara en kalendermässig mardröm att involvera alla berörda deltagare upprepade gånger för att testa och verifiera att mjukvaran fungerar korrekt. De har under åren sett olika attityder gentemot testaktiviteter. Företaget har dock genom sin styrande funktion uttalat att agila metoder och intensiv testning är den praxis man skall arbeta efter. De anser detta som tillräckligt stöd för testledarna lägre ner att driva på att testaktiviteten prioriteras, oavsett vad projektledare och utvecklare anser. De ser att vissa av deras utvecklare har kommit över tröskeln och ser vinsten i att faktiskt skriva sina testfall innan de börjar producera programkod, likt metodiken i testdriven utveckling. Andra utvecklare vill arbeta i gamla mönster där de skriver programkoden först och sedan, tvingas att skriva sina tester för att verifiera att det fungerar. Detta arbetssätt anses vara svårare och koden blir inte testbar på samma sätt.

"Sedan finns alltid de som inte tror på nyttan av testaktiviteter överhuvudtaget".

Sedan finns alltid de som inte tror på nyttan av testaktiviteter överhuvudtaget. Men de ser ändå en positiv trend och en ökad mognadsgrad, där fler och fler inser vinsten med testaktiviteter. De anser att utvecklaren alltid ska känna sig trygg i sitt handlande vilket uppnås genom säkerställning av att systemmiljön fungerade korrekt före förändringen. Utvecklaren kan snabbt fastställa hur defekter och fel uppstått, och backa till det senaste fungerande tillståndet. De anser att automatiserade tester skapar trygghet och kontroll, genom att försäkra sig på att systemet i sin helhet är korrekt.

"Vinsten med testdriven utveckling blir begränsad om man inte använder den fullt ut"

I dagsläget ser de ett problem med den nuvarande integrationsplattformen IBM WebSphere Message Broker som tillhandahåller ett testverktyg, där enhetstester av isolerade delar programkod inte stöds. Då enhetstester är en central del av det testdrivna utvecklingssättet kan de inte, som det ser ut idag, dra full nytta av de positiva aspekterna som den testdrivna utvecklingsmetoden medför.

4.4 Företag 4

Företag 4 är verksamt inom IT & Telekombranschen. De arbetar efter utsatta standarder de har vid testarbete vid integrationsplattformar. Vår respondent har tidigare haft flera roller rörande testarbete av integrationsplattformar och mjukvara, har även varit med och utvecklat deras standarder och rutiner rörande test.

Företaget har genom åren identifierat ett antal attityder rörande testarbete av integrationsplattformar. Där projektledare är alldeles för optimistiska när de beräknar tidsåtgången för projekten, och att de genomgående testar för lite. Detta ser de som en regel snarare än ett undantag och gäller för alla aspekter av testarbete, från enhetstester som utförs av utvecklare till funktionstester och acceptanstester som utförs av verksamheten. En konsekvens av den optimistiska beräkningen de ser är att tid som avsatts för testaktiviteter stjäls i förmån för andra när leveransdatumet börjar närma sig. En annan konsekvens de ser av att de testar för lite är att defekter och fel som hade kunnat identifieras och åtgärdats före leverans följer med efter leveransen, där de enligt företaget är många gånger dyrare att åtgärda. De menar att desto tidigare defekter och fel åtgärdas, desto lättare och mindre kostsamt blir det. Man slipper då hela processen att gå via testmiljöer och involvera slutanvändare för att hitta felen. De menar dessutom att många integrationer är affärskritiska och att inte verifiera dess funktion kan medföra stora risker och kosta miljontals kronor i timmen, om produktionen skulle stå still.

"Man måste värdera kostnaden kontra risken för om det inte skulle fungera".

Företaget talar om riskhantering som en utgångspunkt för att bestämma testinsatsen. De menar att en testplan/teststrategi alltid bör specificeras där olika risker analyseras och på så sätt skapa beslutsunderlag för testinsatsen. Man måste ställa sig frågan; Vad händer om något inte fungerar eller blir felaktigt? Samtidigt belyser de svårigheterna i att risksätta och bedöma testinsatsen. En

metodik som de använder är att bygga upp riskmodeller utifrån komplexiteten hos integrationslösningen och risken (hur affärskritisk är integrationens funktion är). Om det finns stora risker relaterade till integrationslösningen läggs extra tid och resurser på test och kvalitetssäkring. De anser att tester av icke-funktionella krav planeras och utförs också generellt i för låg utsträckning. De beaktar viktiga frågor; Kan hårdvaran hantera en viss volym? Är systemet skyddat mot intrång? Följden av bristfällig testning av icke-funktionella krav kan enligt företaget leda till att man får förändra sin lösning sent i utvecklingsprocessen. De ser att test av funktionella och icke-funktionella krav missas ofta på grund av bristfällig kompetens och förståelse hos projektledaren och utvecklingsgruppen.

"Man kan inte lägga alltför mycket på test, den tiden eller pengarna finns inte. Det är good enough som gäller".

Även om företaget beaktar risker relaterade till att inte testa och bygger riskmodeller, är det vanligt förekommande i de olika projekten att de endast skriver good case testfall, för att verifiera att de beställda funktionerna fungerar korrekt. Detta istället för att kombinera good och bad cases för att exponera svagheter i systemet. Ibland räcker inte tiden till på grund av förseningar och dåliga tidsestimat. Då måste testledaren flagga för att de bara hinner testa vissa delar och de tvingas att produktionssätta systemet med vetskapen att vissa delar inte testats eller testats tillräckligt. Enligt företaget är det emellertid ovanligt att testledaren gör detta vilket utgör risker.

"Vissa individer är kvalitetsmedvetna i grunden och andra vill bara leverera så snabbt som möjligt".

Företaget anser att metodiken för utveckling och testarbete till stor del påverkar den slutliga kvaliteten, genom att den påverkar testinsatsen. Företaget anser det vara viktigt att tillhandahålla en uttalad testprocess och riktlinjer för säkerställa att testinsatsen analyseras och värderas objektivt i relation till riskerna. Utan styrning avgörs testinsatsen av den enskilde projektledaren. De menar att vissa individer är kvalitetsmedvetna i grunden och andra vill bara leverera så snabbt som möjligt. Företaget trycker på vikten av att utvecklarna levererar testdokumentation såsom testfall och testfiler. Inte bara för att kunna återanvända dessa vid framtida kodförändringar utan som bevis att de faktiskt testat och uppnått en högre testtäckning. De

känner till testdriven utveckling och det krav som arbetssättet ställer på utvecklarna att faktiskt skriva testfall innan produktionskod. Företaget belyser vikten av dokumentera testtillgångar med en tidigare erfarenhet rörande ett större migreringsprojekt av en integrationsplattform till en nyare produktversion; IBM WebSphere Message Broker är en central produkt i integrationsvärlden och som det ibland kommer uppdateringar till vilket skapar behov av migrering. Incitamenteten för migreringen var att den befintliga licensen innehöll för få funktioner, men framför allt skulle programversionen sluta supporteras av IBM. Projektet ansågs som stort och innefattade nära 200 integrationer.

”Man tvingades initiera ett omfattande testprojekt för att testa igenom i stort sett alla integrationer”.

Eftersom man tidigare inte hade arbetat strukturerat med test genom att lagra testfiler och testfall, saknades kontroll över programkoden. Man tvingades initiera ett omfattande testprojekt för att testa igenom i stort sett alla integrationer. De involverade verksamheten och de olika systemen som berörde integrationerna. Ett stort problem var att hitta verksamhetsfolk som inte bara var tillgängliga utan också hade kunskap om systemen. Stora ledtider i väntan på lämpligt verksamhetsfolk gjorde att projektet tog väldigt lång tid att genomföra och kostande mer pengar än estimerat. Man valde tillslut att inte uppdatera till den nyare versionen eftersom arbetsinsatsen varit kostsam och långdragen.

5. Analys

Vi ville med vår uppsats undersöka hur testarbete bedrivs rörande utvecklingen och förändring av integrationsplattformar. I detta ingick att undersöka vilka attityder samt svårigheter som finns relaterade till testarbetet. Vi vill dessutom ta reda på vilka tankar företag har kring testdriven utveckling vid utveckling av integrationsplattformar. Vår frågeställning var, som tidigare nämnts *“Vilka övergripande utmaningar finns rörande testarbete vid utvecklings- och förändringsarbete av integrationsplattformar?* I detta avsnitt kommer vi att analysera de svar som vår empiriska studie resulterade i mot bakgrunden i vårt teoretiska ramverk.

5.1 Attityder samt svårigheter relaterade till testarbetet

Efter att varit ute och intervjuat våra respondenter kan vi konstatera att det finns en genomgående positiv inställning gentemot testarbete, alla de företag vi intervjuat hade uttalat från ledningsnivå att testaktiviteter måste prioriteras. Detta stämmer således överens med Iacob & Constantinescu (2008) studie, vilka menade att test och kvalitetssäkring är bland den högst prioriterade frågorna hos chefer för mjukvaruutveckling. Trots den genomgående positiva inställningen som respondenterna har gentemot test, ser vi att det inte är självklart att övriga medarbetare delar samma uppfattning. Detta i enlighet med Iacob & Constantinescu (2008) som menar att testarbetet ofta är en ovälkommen praxis. De individer som intervjuats har haft lång erfarenhet med och besitter stor förståelse rörande test och mjukvaruutveckling. De har sett hur test kan utgöra skillnaden mellan framgång eller misslyckade för hela projekt. Vi ser ett mönster i en vilja att öka medvetenheten hos alla parter i utvecklingsprojekten, om vilka konsekvenser som bristfällig testning faktiskt kan medföra. Både företag 1, 3 och 4 betonar att kostnaden för att åtgärda defekter och fel ökar markant efter lansering. Detta stöds av Iacob & Constantinescu (2008) som menar att kostnaden för att åtgärda defekter och fel i mjukvaran kan bli upp till hundra gånger större än om de skulle upptäckts under utvecklingsprocessen. Tre av fyra organisationer har i nuläget arbetat fram en generell testprocess, medan företag 1 inte gjort detta. Företag 1 ser ett tydligt samband mellan testarbete och kvalitetssäkring. Samtidigt ser de svårigheter med att etablera en fungerande testprocess och följa upp den. De anser att det inte bara krävs en teknisk infrastruktur som stödjer detta, utan också standardiserade arbetssätt som behandlar processen kring testning. Vår studie pekar på att styrning är viktigt och om man inte inte har ett standardiserat ramverk med riktlinjer kring arbetssätt finns det risker i den fria tolkningen. Våra respondenter menar att vissa individer är kvalitetsmedvetna i grunden och andra vill bara leverera så snabbt som möjligt. Ett problem är att det blir en subjektiv bedömning av projektledaren när denne ska bedöma behovet och insatsen av test.

Vi ser en tendens till nonchalans i den testinsats som görs hos samtliga företag, oavsett testprocess. Alla företag vi talat med testar endast huvudfunktioner och använder sig av giltiga värden när de utför system och funktionstester, företag 4 kallar det för “good enough”, företag 3 kallar det för “happy flow” de andra säger “good case”. Detta kan medföra att defekter och fel som potentiellt hade kunnat identifieras under utvecklingsprocessen då istället identifieras istället

av slutanvändaren efter lansering. Vi anser att de också borde stressa systemet för tester som innefattar inmatning av felaktiga värden och sedan utvärdera hur systemet hanterar dessa. Vårt påstående stöds av också SWEBOK:2005 som beskriver vikten av att testa på olika nivåer och med en variation av testtekniker. Företag 1 och 2 är överens om vikten att inte bara testa huvudfunktioner med giltiga värden, men det räcker inte att endast vara medveten om problemet utan åtgärder måste vidtas för att uppnå önskad effekt.

Företagen tyckes avsätta tid för test i de senare faserna i projekten, precis innan lansering. Företag 2 menar att omfattningen i de tidigare faserna skapar en försening genom hela projektet och det grundar sig i dåliga initiala estimeringar för projektets tidsåtgång och resursbehov. Samtidigt ger företag 4 en liknande beskrivning med dålig tidsestimering som upphov till att avsatt tid för testaktiviteter stjäls i förmån till att kunna leverera inom avtalad tid. Amland (2000) styrker detta genom att hävda att när det så småningom kommer till testning i de senare projektfaserna, är tiden tills leveransen extremt kort och det finns ingen budget kvar till tillräcklig testa alla funktioner. Vi vill därför belysa om svårigheten med att avsätta tid för testaktiviteter i en verksamhet där projekten arbetar under begränsade resurser. Vi anser att man borde använda sig av riskbaserad testning för att hantera de förutsättningar som ges och genom ett strukturerat tillvägagångsätt för att bestämma vad som är viktigast att testa före lansering. Detta är i enlighet med Alam & Khan (2013) som menar att utvecklingsprojekt arbetar ofta under begränsade resurser i form av tid, budget och människor. Detta stöds av Amland (2000) som menar att idén om riskbaserad testning är att fokusera på testaktiviteter och spendera mer tid på att testa kritiska funktioner.

5.2 Tankar gentemot testdriven utveckling

Samtliga av de intervjuade respondenterna kände väl till och ser både för- och nackdelar med testdriven utveckling (TDD). Detta handlade om att få en förståelse för respondenternas tankar kring TDD som utvecklingsmetod, deras svar grundar sig i vad de gör idag och hur TDD skulle kunna medföra fördelar. De samlade intrycket vi fick var att de för- och nackdelar respondenterna beskrev under våra samtal återspeglar också de för- och nackdelar som beskrivs i vårt teoretiska ramverk.

Företag 3 ser att testdriven utveckling kan skapa förbättrad effektivt. Men resonerar som så att desto tidigare defekter och fel kan identifieras desto lättare och billigare är de att hantera. Detta påstående delas med Företag 4 som säger att man genom TDD kan minimera risken för att slutanvändaren skall behöva upptäcka felet. Detta stämmer överens med Williams & George (2004) studie som hävdar att TDD's effektivitet i att förhindra och eliminera defekter, minskade resurs- och tidsåtgång vid felsökning och förändringsarbete, kompenserar för den extra tid som går åt att skriva och exekvera testfall och utföra enhetstester.

Företag 1 ser en fördel i att ha möjlighet att bygga upp en testbank av tidigare genomförda och godkända testfall för att sedan återanvända dessa vid framtida förändringar i så kallade regressionstester. Underhållsfixar och mindre kodförändringar under förändringsprocessen kan vara nästan 40 gånger mer utsatta för fel och defekter än nyutveckling, menar Williams & George (2004). Om testfall saknas så har utvecklaren mindre kontroll och vet inte om en kodförändring av funktion bryter funktionen hos en annan (Nagappan et al, 2008). Företag 4 ser konceptet av en testbank som en trygghet för utvecklarna genom att kunna säkerställa att de delar som påverkas av förändringsarbetet tidigare fungerat och verifiera att de fungerar även efter förändring. En fördel alla företag lyfter fram hos är att genom att bli tvingad att skriva testfall före produktionskod har man där försäkrat sig om att testfall skrivs, och kan redovisas. Williams & George (2004) beskriver fördelen att skapa testtillgångar i form av testbanker och menar att eftersom testfall för alla enheter som skall testas skrivs före själva koden gör att tester faktiskt skrivs, detta ger värdefulla resurser för projekten. Detta påstående stärks av Larman (2004) som menar att testdriven utveckling genererar en stor mängd historiska data i form av enhetstester, vilka kan lagras i en så kallad testsvit och återanvändas under utvecklings- och förändringsprocessen för att säkerställa att kodförändringar inte orsakat varken nya eller gamla defekter i systemet.

Efter vår intevju med företag 1 kan vi se en tendens till övertro till TDD, då respondenten endast hade positiva tankar och antaganden om metoden. Företag 2 och 3 varnar att förlita sig blint till en specifik metod kan skapa en känsla av falsk trygghet. Företag 2 menar att även om man strukturerat kör regressionstester av enhetstester på lågnivå måste man också genomföra exempelvis integrationstester på högre nivå i efterhand. När koden klarat på lågnivå, de tester

den ställts inför behöver det nödvändigtvis inte innebära att interaktionen mellan komponenterna fungerar. Vi anser att man bör variera de testnivåer som tester genomförs på och dessutom variera testtekniker och testa för specifika mål, exempelvis stresstest men också testa för att verifiera funktionaliteten. Denna föreställning stöds av Nagappan et al (2008) som menar att ett stort antal passerade enhetstester kan ge en falsk känsla av trygghet, vilket resulterar i färre och varierande testaktiviteter.

De övriga företagen såg till skillnad från företag 1 inte bara vinster utan även svårigheten i tillämpning en av metoden. De övriga företagen inser att TDD har en lång inlärningskurva samt att metoden måste genomsyra hela sättet att bedriva programmering på. Dogsa & Batic (2011) stärker detta genom tidigare studier och påstår att TDD är en svår utvecklingsmetod att implementera, och att det kräver en viss inställning till arbetet. Vår uppfattning är att människor har svårt att bryta gamla mönster och anamma nya arbetssätt om det inte själva kan se tydliga fördelar. Även om TDD skulle implementeras är det därför inte självklart att utvecklarna skulle ta till dig arbetssättet och skriva testfall före produktionskod. Först när alla involverade deltagare anpassat sitt mind-set kan TDD nå sin fulla potential.

6. Slutsats

I följande avsnitt drar vi slutsatser kring vår empiri och teori för att försöka svara på vår frågeställning *"Vilka övergripande utmaningar finns rörande testarbete vid utvecklings- och förändringsarbete av integrationsplattformar?* Vi valde att avgränsa oss till testarbete kring mjukvaruutveckling av integrationsplattformar.

Vår bild av hur testaktiviteter genomfördes har förändrats under arbetets gång genom att vi anskaffat inte bara kunskap från teorier utan också fått ta del av ovärderlig kunskap av branschfolk med lång erfarenhet. Den övergripande utmaningen är att förstå vilken innebörd och betydelse test har för slutprodukten, denna förståelse tror vi kan ökas genom att man involverar och utbildar projektdeltagarna i vilka aktiviteter som de olika testfaserna utgörs av och vilken bidragande effekt de har till slutprodukten.

En utmaning finns med att uppnå ett strukturerat testarbete om en standardiserad testprocess saknas i organisationen. Detta grundar sig i projektstyrande individers egna intressen, mål och uppfattningar om vad som bör prioriteras i ett projekt med ofta hårt begränsade resurser. Om inte individen kan se konsekvensen av bristfällig testning kommer det inte heller prioriteras.

Vårt resultat har visat att de praktiska för- och nackdelar med testdriven utveckling som beskrivits i tidigare litteratur rörande testdriven utveckling, har i denna uppsats styrkts. De ovanstående utmaningarna tror vi testdriven utveckling kan ha positiv inverkan på. Testdriven utveckling är ett bra sätt att faktiskt säkerställa att testfall skrivs. Dock tror vi att det finns brister i att endast förlita på en testmetod eller teknik. Mjukvarutestning bör ske på olika nivåer och med en variation av tekniker för att åstadkomma en heltäckande räckvidd.

7. Källförteckning

Alam, M. M., & Khan, A. I. (2013). *Risk-based Testing Techniques: A Perspective Study*. Risk, 65(1).

Amland, S. (2000). *Risk-based testing: Risk analysis fundamentals and metrics for software testing including a financial application case study*. Journal of Systems and Software, 53(3), 287-295.

Beck, K. (2003) *Test-Driven Development: By Example*. 1. Addison-Wesley Professional

Bernard, H. R. (2006). *Research methods in anthropology: Qualitative and quantitative approaches*. 4. Altamira press.

Dogsa. T. & Batic. D (2011) *The effectiveness of test-driven development: an industrial case study*. Volym 19, Nummer 4, pp. 643 - 661

Erdogmus, H. et al. (2005) *On the Effectiveness of the Test-First Approach to Programming.* Volym 31, Nummer 3, pp. 226 - 237

Fenner, J. *Enterprise Application Integration Techniques*. http://www.cs.ucl.ac.uk/staff/ucacwxe/lectures/3C05-02- 03/aswe21-essay.pdf, hämtad 2013

Ganesh, N., & Thangasamy, S. (2012). *New Agile Testing Modes*. Information Technology Journal, 11, 707-712.

Iacob, I. & Constantinescu, R. (2008) *TESTING: FIRST STEP TOWARDS SOFTWARE QUALITY*. Volym 3, Nummer 3, pp. 241 - 253

ISO/IEC TR 19759:2005(E), *Software Engineering — Guide to the Software Engineering Body of Knowledge* (SWEBOK). Chapter 5, Software Testing

Khan, R.A. & Choudhary, K.R. (2011) *Software Testing Process: A Prescriptive Framework.* Volym 36, Nummer 4, pp. 1 - 5

Klein, H. K., & Myers, M. D. (1999). *A set of principles for conducting and evaluating interpretive field studies in information systems*. MIS quarterly, 67-93.

Larman, C. (2004) *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development.* 3. Pearson Education, Inc

Linthicum, S.D. (1999) *Enterprise Application Integration*. 1. Addison Wesley Professional

Miles, M. B., & Huberman, A. M. (1994). *Qualitative data analysis: An expanded sourcebook*. 2 ed. Sage Publications, Incorporated.

Nagappan, N. et al. (2008) *Realizing quality improvement through test driven development: results and experiences of four industrial teams*. Volym 13, Nummer 3, pp. 289 - 302

Nikolik, B. (2012) *Software quality assurance economics*. Volym 54, Nummer 11, p. 1229

Overby, E., Bharadwaj, A., & Sambamurthy, V. (2006). *Enterprise agility and the enabling role of information technology*. European Journal of Information Systems, *15*(2), 120-131.

Papazoglou, M.P. & Ribbers, P.M.A. (2006) *e-Business: Organizational and Technical Foundations.* 2. John Wiley & Sons Ltd

Positivism. http://www.ne.se/lang/positivism, Nationalencyklopedin, hämtad 2013-05-14.

Schmidt, J.G. & Lyle, D. (2010) *Lean integration : an integration factory approach to business agility*. 1. Pearson Education, Inc

Williams, L. & George, B. (2004) *A structured experiment of test-driven development*. Volym 46, Nummer 5, pp. 337 - 342

Bilaga 1 Intervjufrågor

Bakgrund

- Tidigare roller
- Nuvarande roll
- Dina arbetsuppgifter.

Frågor kring testarbete vid utveckling av integrationsplattformar

- Beskriv er testprocess
- Vad avgör graden av testtäckningen?
- Vilka delar är de mest kritiska att testa för att tillräckligt testa och kvalitetssäkra en integrationsplattform?
- Lärdomar kring testarbete efter att ha jobbat med integrationsutvecklingsprojektet?

Tankar kring testdriven utveckling

- Incitament med att arbeta strukturerat med test vid utveckling av integrationsplattformar?
- Vad talar emot att arbeta strukturerat med test vid utveckling av integrationsplattformar?
- Vilka insatser krävs för att uppnå ett faktiskt värde av testdriven utveckling?