HÖGSKOLAN
DALARNA

Examensarbete

Kandidatuppsats

Testverktyg för test av mobila applikationer

En jämförande fallstudie

Testing tools for test of mobile applications
A comparative case study

Författare: Mirjami Olsson & Erik Norman Uhlin
Handledare: Ulrika Artusson Wissa
Examinator: Bo Sundgren
Ämne/huvudområde: Informatik
Kurskod: IK2017
Poäng: 15 hp
Examinationsdatum: 2015-06-09

Vid Högskolan Dalarna har du möjlighet att publicera ditt examensarbete i fulltext i DiVA. Publiceringen sker Open Access, vilket innebär att arbetet blir fritt tillgängligt att läsa och ladda ned på nätet. Du ökar därmed spridningen och synligheten av ditt examensarbete.

Open Access är på väg att bli norm för att sprida vetenskaplig information på nätet. Högskolan Dalarna rekommenderar såväl forskare som studenter att publicera sina arbeten Open Access.

Jag/vi medger publicering i fulltext (fritt tillgänglig på nätet, Open Access):

Ja ☒ Nej ☐

HÖGSKOLAN DALARNA

EXAMENSARBETE, Grundnivå 2 i Informatik

Ämne	Reg nr	Omfattning
Informatik, Grundnivå 2	08/2015	15 hp
Namn	Månad/År	
Mirjami Olsson Erik Norman Uhlin	Juni 2015 Handledare: Ulrika Artusson Wissa Examinator: Bo Sundgren	
Företag/Institution	Handledare vid företaget/institutionen	
Triona AB	Cecilia Grahn & Erica Magnusson	
Titel		
Testverktyg för test av mobila applikationer		
Nyckelord		
automatiserade tester, test av mobila applikationer, testverktyg		

Sammanfattning

Användandet av mobila applikationer har växt radikalt de senaste åren och de samverkar med många system. Därför ställs det högre krav på kvaliteten och att applikationen ska anpassas till många olika enheter, operativsystem samt plattformar. Detta gör att test av mobila applikationer blivit viktigare och större. Detta arbete har bedrivits som en jämförande fallstudie inom området test av mobila applikationer samt testverktyg. Syftet har varit att beskriva hur testning av mobila applikationer sker idag vilket gjorts genom litteraturstudier och intervjuer med IT-företag. Ett annat syfte har varit att utvärdera fyra testverktyg, deras för- och nackdelar samt hur de kan användas vid testning av mobila applikationer och jämföras mot manuell testning utan testverktyg. Detta har gjorts genom att skapa förstahandserfarenheter baserat på användandet av testverktygen. Under arbetet har vi utgått från mobila applikationer som vi fått tillgång till av Triona, som varit vår samarbetspartner.

Idag finns många olika testverktyg som kan användas som stöd för testningen men få företag har implementerat något eftersom det kräver både tid och kompetens samt valet av testverktyg kan vara svårt. Testverktygen har olika för- och nackdelar vilket gör att de passar olika bra beroende på typ av projekt och applikation. Fördelar med att använda testverktyg är möjligheten att kunna automatisera, testa på flera enheter samtidigt samt få tillgång till enheter via molnet. Utmaningarna är att det kan vara svårt att installera och lära sig testverktyget samt att licenserna kan vara dyra. Det är därför viktigt att redan innan implementationen veta vilka tester och applikationer testverktygen ska användas till samt vem som ska använda det. Utifrån vår studie kan slutsatsen dras att inget testverktyg är helt komplett men de kan bidra med olika funktioner vilket effektiviserar delar av testningen av mobila applikationer.

DALARNA
University College

DEGREE PROJECT, Undergraduate level 2 in Informatics

Subject	Reg number	Extent
Informatics, Undergraduate level 2	08/2015	15 ects
Names Mirjami Olsson Erik Norman Uhlin	Month/Year June 2015	
	Supervisor: Ulrika Artusson Wissa Examiner: Bo Sundgren	
Company/Department Triona AB	Supervisor at the Company/Department Cecilia Grahn & Erica Magnusson	
Title Testing tools for test of mobile applications		
Keywords testing tools, mobile application testing, automated testing		

Abstract

The use of mobile applications has grown dramatically in recent years and they interact with many systems. Therefore the requirement on the quality is higher and the application must often be adapted to many devices, operating systems and platforms. This means the testing of mobile applications becomes more and more important. This study has been conducted as a comparative case study in the field testing of mobile applications and testing tools. The aim has been to describe how the testing of mobile applications takes place today which have been made through literature studies and interviews with IT-companies. Another aim was to evaluate four testing tools, their advantages and disadvantages and how they can be used in the testing of mobile applications and compared to manual testing without testing tools. This was done by using and experience the testing tools. During the study, we have used mobile applications that we got access from Triona, who has been our partner.

Today there are many different testing tools that can be used to support testing, but few companies have implemented these because it requires both time and expertise as well as the selection of testing tools can be difficult. Testing tools have different advantages and disadvantages which make them fit differently depending on the type of project and application. Advantages of using testing tools is the ability to automate, test on multiple devices simultaneously and get access to physical devices via the cloud. The challenges are that it can be difficult to install and learn the testing tool and that the licenses can be expensive. It is therefore important that even before implementation to know which tests and applications the testing tools will be used in and who will use it. Based on our study we can conclude that no testing tools are complete but they may contribute with different features which streamlines parts of the mobile applications testing.

Förord

Detta examensarbete har genomförts som den avslutande delen på det Systemvetenskapliga programmet på Högskolan Dalarna. Området till examensarbetet har valts då det känts både aktuellt och intressant.

Vi vill tacka vår samarbetspartner Triona i Borlänge för denna möjlighet och för ett bra samarbete. Ett speciellt tack riktas till handledarna Cecilia Grahn och Erica Magnusson hos Triona för deras stora hjälp.

Vi vill även tacka vår handledare Ulrika Artusson Wissa på Högskolan Dalarna samt övriga handledare och våra medstuderande för all feedback och stöd.

Borlänge, 9 juni 2015

Mirjami Olsson
Erik Norman Uhlin

Innehållsförteckning

Begreppsdefinitioner

API	Application Programming Interface är en specifikation av hur olika applikationsprogram kan använda och kommunicera med en specifik programvara (Wikipedia, 2015).
APK-fil	APK är ett filformat som används för att installera och distribuera Android applikationer (Wikipedia, 2015).
Cordova	Cordova är ett ramverk för utveckling av hybrid-applikationer från Apache (Wikipedia, 2015).
Cross-Platform	Cross-Platform innebär att applikationer är byggda på samma kodbas, och sedan kompileras till olika native format (Kohl, 2013).
Emulator	En emulator är en hårdvara eller mjukvara vars syfte är att efterlikna annan hårdvara eller mjukvara, det möjliggör även att programvara kan köras på emulatorer (Wikipedia, 2015).
IDE	IDE (Integrated Development Environment) är en programvara eller utvecklingsmiljö som vanligtvis tillhandahåller textredigerare, kompilator samt debugger för att underlätta vid programmering (Wikipedia, 2013).
IPA-fil	IPA-filer är komprimerade filer som innehåller IOS applikationer (Wikipedia, 2015).
Jailbreaking	Jailbreak gör det möjligt att installera och använda program från tredje part, det vill säga mjukvara som inte är godkända av Apple på en IOS enhet (Wikipedia, 2014).
JDK	Java Development Kit är en utvecklingsmiljö som innehåller olika komponenter och programmeringsverktyg (Wikipedia, 2015).
Jenkins	Jenkins är ett Open Source program skrivet i Java som övervakar exekvering av upprepade mjukvaruprojekt. (Jenkins, 2015)
Molntjänst	Molntjänster är IT-tjänster som tillhandahålls över Internet, exempel på molntjänster är Dropbox, iCloud, Office 365 och Flickr (Wikipedia, 2015).
Open Source	Open Source, eller öppen källkod, innebär att källkoden är tillgänglig att använda, läsa, modifiera och vidaredistribuera för vem som helst (Wikipedia, 2015).
SDK	Software Development Kit är en samling utvecklingsverktyg för utvecklare så de kan bygga applikationer mot exempelvis ett specifikt ramverk, plattform, operativsystem eller programpaket (Wikipedia, 2014).

1 Inledning

I kapitlet beskrivs bakgrunden till examensarbetet som mynnar ut i problemformulering och syfte inkluderande forskningsfrågorna. Vi presenterar även samarbetspartnern, mål, avgränsning, tidigare forskning samt till vilka rapporten riktar sig till.

1.1 Bakgrund

Testning är ett stort och viktigt område under utveckling av IT-system med höga kvalitetskrav. Testarbetet beskrivs som lyckat när användarna får ett välfungerande system vilket leder till ökad vinst och minskade kostnader. Tester kan göras både manuellt och automatiserat. Manuella tester utförs av människor och resultatet skrivs i textform. Automatiserade tester innebär att många tester kan genomföras på kort tid och de körs automatiskt i ett testverktyg. (Eriksson, 2008)

I dagens samhälle samverkar många system med mobila applikationer. Användandet av mobila applikationer har växt radikalt de senaste åren vilket leder till att kvaliteten är ett av de högst prioriterade målen för företag inom apputveckling. (Kipar, 2014) Detta för med sig nya utmaningar för utvecklingen av mobila applikationer. Några av de största utmaningarna för IT-företagen är de snabba tekniska förändringarna inom mobila enheter, alla olika plattformar, storlekar och enheter som applikationen måste anpassas för samt testningen av dessa. (Mao & Xin, 2014) Ytterligare något som ska tas hänsyn till är att en mobil har jämfört med datorer, begränsad kapacitet och tillgång till nätverk beroende på var mobilen befinner sig. (Hughes Systique Corporation, 2013)

Ovannämnda utmaningar är faktorer som gör även testprocessen extra viktig vilket därmed blir utmanande och tidskrävande vid manuell testning av mobila applikationer utan testverktyg. För att underlätta testningen finns idag många olika testverktyg anpassade för utvecklingen av mobila applikationer. Dessa testverktyg kan användas både för manuella samt automatiserade tester. Hughes Systique Corporation (2013) beskriver att automatiserade tester med testverktyg av mobila applikationer har många fördelar så som ökad effektivitet, fler tester på kortare tid, förbättrade regressionstester samt repetitiva testprocesser. Kochhar, Thung, Nagappan, Zimmermann & Lo (2015) har gjort en empirisk studie med drygt 200 utvecklare där 65 % av dem använde sig av testverktyg. I studien framkom det att utvecklarna såg testverktyg som ett bra hjälpmedel som underlättar och effektiviserar testningen. Men det fanns även utmaningar med dessa såsom att lära sig själva testverktygen och att vissa applikationer har anpassad funktionalitet som var svår att testa med testverktyg. Även Kipar (2014) beskriver oerfarna eller tillfälliga testare, knappa tidsresurser eller om testarna inte är bekanta med testverktyget som ytterligare utmaningar.

Eriksson (2008) menar att testverktygen passar olika bra till olika typer av tester. För att testverktygen ska effektivisera och underlätta utförandet av de olika testerna är det viktigt att testverktyget täcker behoven. Detta gör att valet av testverktyg blir väldigt viktigt. Plotytsia (2014) på TestLab4Apps presenterar en guide på elva steg som kan användas som stöd i processen att välja rätt testverktyg för sina ändamål. Denna guide används och beskrivs i detta arbete.

Vår samarbetspartner för arbetet är IT-företaget Triona i Borlänge som bland annat utvecklat ett antal mobila applikationer. I dagsläget använder de inget testverktyg vid testningen av mobila applikationer och antalet enheter som applikationerna stödjer är begränsade.

1.2 Problemformulering

Trots att det idag finns väldigt många olika testverktyg så är det få IT-företag som använder sig av dessa. Det kan vara svårt att välja rätt eftersom olika testverktyg har olika för- och nackdelar.

För att underlätta och förbättra testprocessen av mobila applikationer, jämförs olika testverktyg och möjligheten att göra delar av testerna automatiserade. Detta mynnar ut i följande frågeställning med delfrågor:

Hur kan olika testverktyg användas för tester av mobila applikationer?

Delfrågor:
1. Hur utförs testningen av mobila applikationer idag inom IT-företag och vilka utmaningar finns det?
2. Vilka för- och nackdelar finns med testverktygen?
3. Hur kan testningen av mobila applikationer förbättras vid införandet av ett testverktyg samt genom automatisering jämfört med manuell testning utan testverktyg?

1.3 Syfte

Ett syfte med examensarbetet är att beskriva hur testning av mobila applikationer sker idag. Ett annat syfte är även att utvärdera testverktyg, deras för- och nackdelar samt hur de kan användas vid testning av mobila applikationer och jämföra detta mot manuell testning utan testverktyg.

1.4 Mål

Målet med examensarbetet med utgångspunkt från samarbetspartnern Triona är att göra en litteraturstudie kring testverktyg för mobila applikationer samt testverktygens möjligheter till automatisering. Utifrån litteraturstudien jämförs egenskaperna samt för- och nackdelar hos de olika testverktygen för att beskriva möjligheterna att använda sig av dessa. Det görs även en jämförelse på vilka funktioner i testprocessen som kan möjliggöras och förbättras med testverktyg jämfört med manuell testning utan testverktyg.

1.5 Avgränsning

Vi har avgränsat oss till jämförelse av fyra olika testverktyg, som valts utifrån resultatet från litteraturstudien och med hjälp av guiden presenterad av Plotytsia (2014). Testningen är inriktad på funktionella tester på mobila applikationer. Vi har utgått från tre mobila applikationer som vi fått tillgång till från samarbetspartnern.

1.6 Tidigare forskning

Tidigare forskning inom området testverktyg för testning av mobila applikationer som vi under detta arbete tagit stor del av, är metoden av Plotytsia (2014) som beskriver de 11 stegen i en guide för att stödja processen att välja rätt testverktyg. Denna metod har legat som grund för vårt arbete att välja ut och jämföra egenskaperna på testverktyg. Metoden hittades under

litteratursökningen i studien gjord av Kipar (2014). Även den studien har utgått från samma 11 steg för att välja rätt testverktyg. Den har därför haft stor betydelse för vårt arbete då vi genomfört en liknande jämförelse. Studien av Kipar (2014) innehöll endast en teoretisk genomgång av jämförelse mellan de olika testverktygen. Vi har utgått från den men även byggt vidare genom att utöver detta också jämfört och använt dessa testverktyg.

För att ge oss en bakgrund samt överblick av området var den empiriska studien av Kochhar, Thung, Nagappan, Zimmermann & Lo (2015) en stor hjälp då den visar många utmaningar med test av mobila applikationer. Smartbear Software (2014) har gjort en studie med över 1000 utvecklare om på vilket sätt de testar mobila applikationer. Deras insamlade material är från oktober-december 2013. Denna har vi utgått ifrån för att kunna jämföra och reflektera över våra intervjusvar om hur testningen av mobila applikationer ser ut idag.

1.7 Intressenter

Vår samarbetspartner är ett IT-företag som utvecklar mobila applikationer. Det finns många liknande företag som också utvecklar mobila applikationer och som därför kan ses som intressenter för detta arbete. Även konsultbolag som arbetar inom testområdet kan ses som intressenter till arbetet.

Vårt examensarbete bidrar till ämnet Informatik genom att utvärdera testverktyg och deras för- och nackdelar för mobila applikationer. Examensarbetet bidrar även med en jämförelse med hur testprocessen för mobila applikationer ser ut utan testverktyg samt vilka delar som kan möjliggöras och förbättras med testverktyg.

1.8 Samarbetspartner

Samarbetspartner för examensarbetet är Triona i Borlänge som är ett IT-företag grundat 1991 med huvudsaklig fokus på logistik- och transportrelaterad verksamhet. Företaget finns på ett flertal orter i Sverige med huvudkontor i Borlänge, men även i Norge och har sammanlagt ca 140 anställda. Triona erbjuder både egna produkter samt konsulttjänster inom olika områden såsom molntjänster, förvaltning och drift.

2 Teori

Teori kring testområdet, testautomatisering samt mobila applikationer beskrivs. En beskrivning av testverktyg för mobila applikationer samt de utvalda testverktygen som detta examensarbete fokuserar på presenteras. Detta görs för att få en teoretisk referensram som behövs för att kunna förstå området och ha som underlag för arbetet som i sin tur leder till svar på forskningsfrågor.

2.1 Test

Avsnittet innehåller teori om testprocessen, testunderlag och tekniker samt testnivåer.

En definition av test är att exekvera en programvara för se om den uppfyller de specificerade kraven samt för att hitta fel. Test är därför till för att säkerhetsställa att kraven uppfylls men också att kvaliteten på systemet är hög. Om testerna misslyckas kan det få vissa konsekvenser, exempelvis högre utvecklings- och förvaltningskostnader, förseningar vid driftsättning, missnöjda kunder och förlorat anseende (Eriksson, 2008).

2.1.1 Testprocess

Testprocessen behövs för att testarbetet ska bli så effektivt och så väl genomfört som möjligt. Det krävs då att testaren följer de delar som tesprocessen består av vilket är planering, genomförande och uppföljning. De olika delarna beskrivs nedan enligt Eriksson (2008):

Planering
Vid denna del av testprocessen tas bland annat testplan eller teststrategi fram som beskriver vad som ska testas och av vem, när hur och varför. Utöver det innehåller planeringen delar så som: gå igenom och granska kravdokument, bedöma riskerna, ta fram olika testunderlag, testfall, checklistor och testdata samt identifiera testområden.

Genomförande
Vid genomförandet ska testplanen följas. Därför är det viktigt att denna del är färdig och finns dokumenterad. Testunderlagen som togs fram vid planeringen används. Rapportering ska ske löpande samt omtest och regressionstest ska genomföras.

Uppföljning
När testerna är genomförda dokumenteras dessa i en testrapport vilket även bör reflekteras över, vad som gått bra respektive dåligt under testarbetet.

2.1.2 Testunderlag och tekniker

Blackbox, whitebox och ad-hoc är olika testtekniker som kan användas. Med blackbox-testning innebär att testaren inte vet hur systemet är konstruerat inuti medan whitebox innebär det motsatta, att testaren känner till systemets interna struktur. Som testunderlag kan testfall användas. (Eriksson, 2008) Nedan beskrivs ad-hoc samt testfall som är aktuella för detta examensarbete.

Ad-hoc
Ad-hoc testning enligt Eriksson (2008) innebär att testa utan testunderlag som t.ex. testfall. Många fel kan hittas med hjälp av ad-hoc, speciellt vid nyutveckling. Detta på grund av att nya system vanligtvis har fler fel, det är även bra för nya testare att börja med ad-hoc eftersom de då

ser på systemet med nya ögon. En nackdel med ad-hoc är att det kan vara svårt att genomföra exakt samma test igen, det kan därför vara bra att använda något testverktyg som spelar in vad som sker i systemet. Författaren menar att det kan vara lämpligt att använda ad-hoc tester som komplement efter de planerade testerna.

Testfall
Testfall beskriver en situation, funktion eller data som exekveras i programmet för att testa en specifik egenskap. Ett testfall innehåller olika delar eller ”fält”, som exempelvis ID, Rubrik, Förberedelser, Teststeg och Förväntat resultat (Eriksson, 2008).
Ett exempel på hur ett testfall kan se ut ses på figuren nedan:

ID	10
Rubrik	Lägg till ny kund
Förberedelser	Logga in
Teststeg	1. Välj lägg till ny kund funktionen. 2. Ange kundens uppgifter. 3. Klicka på knappen ”Spara”.
Förväntat resultat	Ett meddelande visas att kunden blivit sparad.

Figur 1 - Exempel på ett testfall. Källa: Eriksson (2008) s. 124

2.1.3 Testnivåer

Enhetstester
Enhetstester, även kallat komponenttest, modultest, programtest och unit test, handlar om att testa systemets minsta delar och komponenter på lägsta detaljnivå. Exempelvis kan komponenterna vara HTML-sidor, exe filer, COM-objekt och klasser. De testas mot kraven som är satta i kravspecifikationen. Exempelvis genomförs tester av enskilda funktioner som: spara kund, ändra kund eller olika sökfunktioner, enskilda fält samt tester för en sida eller skärmbild. Dessa tester genomförs främst av utvecklare och sker löpande under utvecklingen, vilket gör att de inte har ett start- eller slutdatum som de andra nivåerna. Testerna kan även automatiseras med hjälp av testramverk som utvecklaren kan använda genom att skriva testskript som exekverar programvaran och kontrollerar att resultatet uppfylls. Det kan exempelvis handla om att kontrollera att ett lösenord har minst 8 tecken och en stor bokstav (Eriksson, 2008).

Systemtest
Systemtest innebär att hela systemet testas, detta sker i en produktionslik miljö. Detta innebär att systemet ska vara färdigt även med kopplingar till andra system. Har systemet många kopplingar till andra system kan dessa tester genomföras på en egen testnivå. Fokus ligger inte på detaljerna och de små funktionerna i systemet eftersom detta sker på en högre detaljnivå än andra testnivåer. Testerna sker vanligtvis av ett testteam och inte av utvecklare. Systemtester är ofta väldigt omfattande och kan ta flera veckor att genomföra och de genomförs oftast av en blandning av funktionella och icke-funktionella testtyper som basers på respektive krav (Eriksson, 2008). Dessa beskrivs nedan:

Funktionella tester
Baseras på de olika funktioner samt egenskaper som är beskrivna i den övergripande designen

men även vid kraven. Exempel på funktionella tester kan vara: lägg till en kund och testa att det går att söka fram kunden med sökfunktionaliteten, eller kolla olika behörigheter för användare i systemet.

Icke-funktionella tester
Sker med hjälp av icke-funktionella testtekniker. Några av dessa är stresstest, prestandatest, integrationstest, användbarhetstest samt säkerhetstest. Prestandatester är aktuellt i detta arbete och innebär att prestandan på systemet testas så att det klarar av de krav som ställts i kraven. Dessa tester kan även kallas lasttest eller volymtest och dessa mäts oftast i svarstid eller transaktionsfrekvens och genomförs med hjälp av testverktyg.

Acceptanstest
Ett högnivåtest som sker precis innan ett system ska tas i bruk. Testet genomförs av slutanvändare och acceptanstestare, och syftet är att godkänna systemet så att det kan tas i drift. Vid acceptanstester skrivs vanligtvis ett lite längre testfall som är mer sammanhängande och med olika aktiviteter. (Eriksson, 2008)

2.2 Mobila applikationer och testverktyg
Avsnittet innehåller teori om mobila applikationer, test av dessa, testverktyg samt testautomatisering.

2.2.1 Mobila Applikationer
Med mobila applikationer menas en programvara skapad för att köras på smarta telefoner, fortsättningsvis kallad mobila enheter. Applikationerna är utvecklade för olika operativsystem, exempelvis IOS, Android & Windows Phone. (Madhushani, De Silva, Madushanka, Malalagama & Manawadu, 2014)

Det finns några olika sätt att utveckla mobila applikationer, de kan vara native-, hybrid- eller Mobila webbapplikationer. Här nedan beskrivs de olika typerna av mobila applikationer.

Native-applikationer
Native-applikationer är utvecklade för en specifik plattform, t.ex. IOS, Android eller Windows (Mao & Xin, 2014). De är skrivna i ett specifikt språk för just den plattformen och kompileras enligt respektive standard. De använder olika utvecklingsverktyg eller SDK där några exempel är: Xcode, Android SDK och Visual Studio (IBM, 2012).

IBM (2012) nämner några för- och nackdelar med native-applikationer. En nackdel med att utveckla native-applikationer är att de inte fungerar på någon annan plattform, vilket gör att det kan ta längre tid att utveckla och det är svårare att underhålla dessa applikationer. Men fördelen är att tillgång fås till all funktionalitet samt djupare förståelse kring API:er och bättre prestanda i applikationerna.

Mobila Webbapplikationer
Dessa är utvecklade med webbtekniker som HTML, JavaScript och CSS och är optimerade för att användas i mobilens webbläsare. De fungerar därför på alla plattformar och enheter men har inte tillgång till all funktionalitet i enheten och plattform (IBM, 2012).

Hybrid-applikationer
Hybrid-applikationer är utvecklade med webbtekniker som HTML, JavaScript och CSS men kombineras även med native-teknik. Denna typ av applikation har tillgång till respektive funktionalitet som finns i enheten och plattformen, men delarna som är gjorda med webbtekniker kan användas på olika plattformar och enheter. Det gör att native-delen och webbdelen kan utvecklas separat. Webbdelen kan antingen användas på en server eller hanteras lokalt på enheten, men är den på en server finns inget offline-stöd (IBM, 2012).

2.2.2 Testning av mobila applikationer
Eftersom applikationer idag blivit mer dynamiska samt användbara och används i betydligt större utsträckning än tidigare, finns det mindre utrymme för misstag och kvalitetskraven är större (Alharthi, 2014). Vid test av mobila applikationer testas inte enbart mjukvaran utan även hur applikationen fungerar när man är i rörelse och är beroende av nätverket. Bland annat så ska enhetens hårdvarufunktioner så som sensorer också tas hänsyn till. (Kohl, 2013) En stor utmaning enligt Kirubakaran & Karthikeyani (2013) är att kunna testa på flera enheter samt vilka enheter som ska väljas. Detta då det exempelvis finns många enheter som har Android som också finns i många olika versioner.

Kirubakaran & Karthikeyani (2013) nämner några olika typer av tester som kan genomföras på mobila applikationer och dessa är:

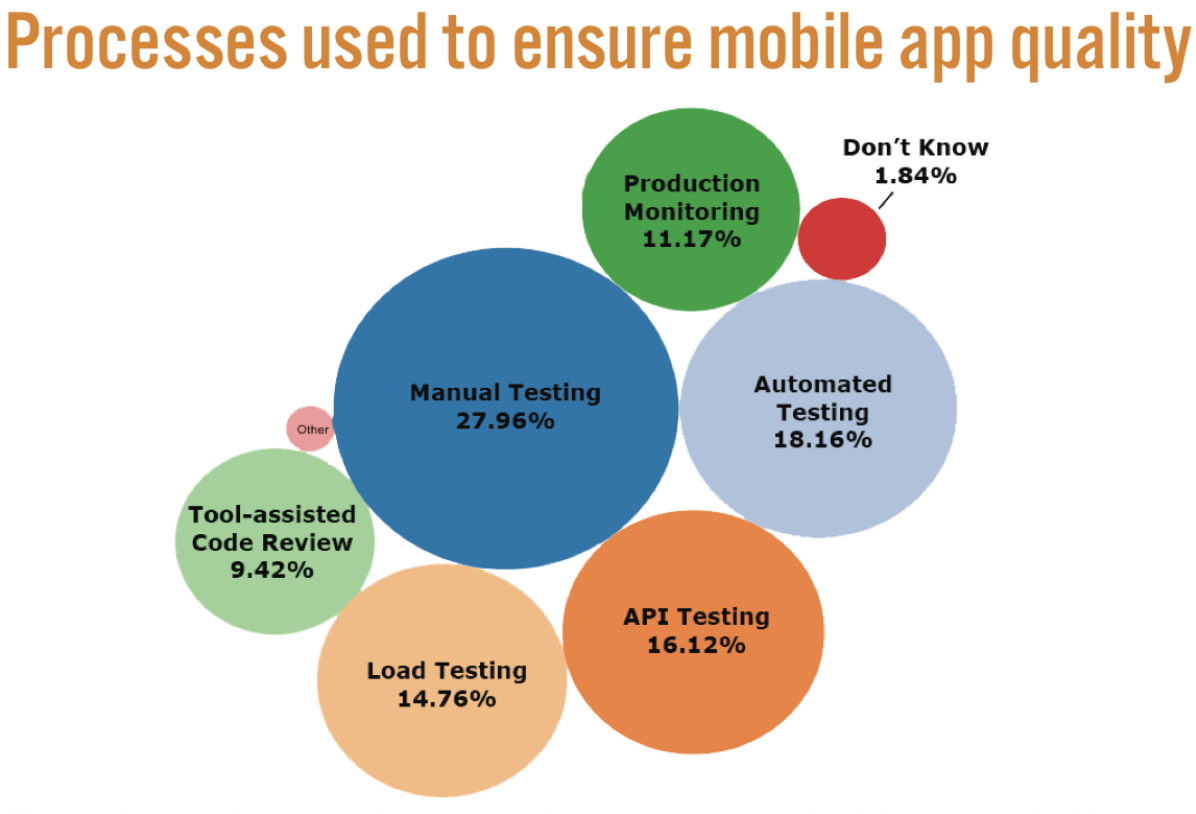
GUI-tester
Testa att alla GUI-komponenter är där de ska på olika enheter och skärmstorlekar. Med GUI-komponenter menas själva användargränssnittet. Även rendering av data, att det sker på korrekt sätt är viktigt att testa. Automatisering av detta vore att spela in skript vad användaren gör för att sedan kunna spela upp dessa på olika enheter.

Prestanda
Eftersom prestandan skiljer sig väldigt mycket mellan olika enheter är det viktigt att analysera detta. Även nätverkets pålitlighet är viktigt att ta med i beräkningar eftersom nätverksuppkoppling inte alltid är tillgängligt.

Säkerhetstester
Eftersom mobila enheter innehåller privat information och uppkoppling sker på många olika nätverk med olika säkerhetsnivåer, blir risker för intrång större. Detta gör att testningen blir extra viktig.

Minne och batteriförbrukning
Testning av minne- och batteriförbrukning för att se och analysera vad som orsakar mest förbrukning. Vidare beskriver Kirubakaran & Karthikeyani (2013) att vid funktionella tester är det viktigt att specificera både applikationens funktionalitet men också den miljö som applikationen ska användas i.

Figuren nedan visualiserar hur tester på mobila applikationer utförs och det går att utläsa hur stor del av testerna görs manuellt eller automatiserat. Den är baserad på en undersökning som Smartbear (2014) gjort om vilka processer som används för att kvaliteten ska bli så hög som möjligt på mobila applikationer. Undersökningen är utförd på omkring 1000 utvecklare, testare och användare. Undersökningen lyfter fram de olika sätten mobila applikationer testas på och visar att manuell test är störst med nästan 28 % medan automatiserade tester står för drygt 18 %.



Figur 2 – Figur över fördelningen av testningen av mobila applikationer. Genomförd av och hämtad från Smartbear (2014)

2.2.3 Testverktyg

Testverktyg förknippas ofta med testautomatisering, att manuella tester körs i ett testprogram automatiskt. Men det finns många andra typer av testverktyg som används för att underlätta testarbetet vid olika tillfällen. Nedan beskrivs fördelar och nackdelar med testverktyg:

Fördelar med testverktyg enligt Eriksson (2008):

- Vissa tester går att genomföra som annars inte skulle fungera, t.ex. prestandatest.
- Fler tester kan genomföras på kortare tid. Automatisering av tester som kommer att köras hela livslängden av systemet sparar mycket tid.
- Hantering av testdata blir betydligt enklare, då testdata kan kopieras och flyttas mellan olika miljöer.
- Bedömningar blir enklare då verktyg tillhandahåller objektiva mätetal. T.ex. för krav- och kodtäckning, systembeteende eller felfrekvens.

Nackdelar med testverktyg enligt Eriksson (2008);

- De är ofta väldigt dyra, speciellt automatiserade testverktyg.
- Det krävs specialkompetens inom programmering av testskript för automatisering.

- Automatiserade tester sparar inte alltid tid men höjer kvaliteten. Det kräver ibland anställning av fler personer som kan hantera verktygen.
- Risk för dåligt underhåll av testskript, eller testskripten kan vara dåligt skrivna. Detta kan leda till att fel inte uppkommer, om samma testskript körs varje gång.

Testverktyg specifika för mobila applikationer har många fördelar så som ökad effektivitet, fler tester på kortare tid, förbättrade regressionstester samt repetitiva testprocesser (Hughes Systique Corporation, 2013). Nackdelar med testverktyg är att det är svårt att lära sig själva testverktygen och att vissa applikationer har anpassad funktionalitet som är svår att testa med testverktyg. Många testverktyg saknar även tillräcklig dokumentation och support. Tekniker som används vid utvecklingen av de mobila applikationerna påverkar hur dessa kan användas i de olika testverktygen vilket också kan vara en nackdel. (Kochhar, Thung, Nagappan, Zimmermann & Lo, 2015)

2.2.4 Testautomatisering

Att omvandla manuella testfall till ett testprogram innebär att dessa kan automatiseras. Detta leder till att många tester kan genomföras på kortare tid. Programmet kontrollerar om resultatet av testet uppfylls och rapporterar det sedan. Testverktyg för testautomatisering innehåller vanligtvis följande delar: Inspelningsfunktion, Redigeringsverktyg, Uppspelningsverktyg och Jämförelse av verkligt resultat med förväntat resultat (Eriksson, 2008).

Enligt Eriksson (2008) kan automatisering delas in i tre nivåer efter hur avancerade testerna som ska genomföras är:

Simpel inspelning/uppspelning
Innebär att testaren utför några olika teststeg som spelas in av verktyget och skapar ett testskript. Detta kan sedan köras igen, men om något ska ändras får testaren spela in ett nytt testfall. En nackdel är att dessa måste uppdateras när systemet ändras, men en fördel är att vem som helst kan skapa automatiserade tester.

Ändring av de inspelade testskripten
Innebär att inspelade testskript ändras med hjälp av ett redigeringsverktyg. Detta gör att exempelvis indata kan ändras. Det ställer dock högre teknisk kompetens eftersom testaren måste kunna läsa och ändra i programkod som testskriptet är skrivet i. Dessa typer av testverktyg är bäst vid system- och acceptanstest.

Tester där testdata hämtas dynamiskt
Den mest komplexa men den mest eftersträvade formen av automatisering. Testdata hämtas dynamiskt för att indata ska variera, det hämtas från exempelvis textfiler, databaser eller Excel filer. Denna nivå ställer högst krav på testarens programmeringskunskaper.

2.3 De valda testverktygen

De testverktyg som examensarbetet fokuserar på presenteras nedan. Testverktygen är valda utifrån litteraturstudien samt med hjälp av guiden från Plotytsia (2014) som presenteras i avsnitt 3.5

2.3.1 Keynote DeviceAnywhere

Keynote DeviceAnywhere är ett molnbaserat testverktyg för testning av både IOS, Android och Windows Phone applikationer. Testerna görs på riktiga enheter, som finns på en server och som fås tillgång till via molnet. I dagsläget finns det 500+ enheter att tillgå. Testerna kan genomföras manuellt eller automatiskt och dessa kan spelas och visas genom video eller skärmdumpar. Testresultaten visas i HTML-sidor med komplett testrapport. Keynote stödjer GUI-baserade kommandon vid skapandet av testskript och kan integreras med andra testramverk. (Keynote, 2015)

En fördel med Keynote är att de har många enheter att välja emellan som fås tillgång till via molnet och behöver inga fysiska enheter. En annan fördel är även att det är en molntjänst som kan nås från hela världen och samma testskript kan användas till de olika plattformarna. Två nackdelar med Keynote är däremot att det blir dyrt om användaren vill få all funktionalitet samt att det blir viss fördröjning vid interaktionen med enheten. (Shlykov, 2013).

2.3.2 AppThwack

AppThwack är ett molnbaserat testverktyg som stödjer testning av HTM5, IOS och Android-applikationer. Testerna görs på riktiga fysiska enheter på en server. Dessa fås tillgång till via molnet och det finns 300+ enheter att välja emellan. Testverktyget kan användas för att testa native- hybrid- samt webbapplikationer och stödjer automatisering av dessa. (AppThwack Overview, 2015)

Applikationerna testas genom att en APK eller en IPA laddas upp på hemsidan så det krävs ingen källkod eller modifiering av koden. Testskripten skapas genom att antingen använda deras AppExplorer eller uppladdning av testskript från ett annat testverktyg som går att integrera. Testskript skapade med AppExplorer kan inte återuppspelas utan skapas av testverktyget. Resultatet presenteras i en full testrapport inkluderandes bilder. Denna testrapport kan delas eller exporteras. AppThwack erbjuder även prestandaöverblick som visas med hjälp av diagram och tabeller. Någon datordriven inmatning stöds inte och det finns ingen unik objektidentifiering. (AppThwack FAQ, 2015) Testverktyget kan användas gratis begränsat till ett antal minuter och finns i köpversion med prissättningen 20-500/månad. (AppThwack Pricing, 2015)

2.3.3 MonkeyTalk

MonkeyTalk är ett testverktyg som kan användas för att testa native- och hybrid-applikationer på fysiska enheter samt emulatorer. Det stödjer både IOS & Android. Verktyget kräver inte Jailbreak och har en integrerad testmiljö där testerna både kan skapas, köras samt förändras. MonkeyTalks egna testskript kan skrivas med deras egna språk alternativt JavaScript. Testerna kan spelas in och testloggen visas i HTML-sidor med skärmdumpar alternativt i en XML-fil. Testverktyget kräver inte någon APK/IPA men däremot modifiering av kod i form av tillägg av bibliotek. (CloudMonkeyMobile, 2015)

MonkeyTalks fördel ligger i att de har en egen IDE med funktioner för inspelningar av testskript samt att de stödjer datordriven inmatning. MonkeyTalks nackdel ligger i saknat stöd för HTML5 webb-applikationer och testerna för IOS kan enbart göras via WiFi. Det går heller inte att testa inbäddade webbsidor med testverktyget. (Kipar, 2014).

2.3.4 Appium

Appium är ett Open Source och Cross-Platform testverktyg. Verktygen kan användas till de olika plattformarna så som IOS, Android samt Windows och testar mobila applikationer som är både native-, hybrid- samt webbapplikationer. Med hjälp av Appiums testverktyg kan automatiserade tester utföras och någon modifiering av källkoden behöver inte göras förutom när testerna görs på IOS enheter. Appium stödjer programmeringsspråken Java, Ruby, Node.js, Python, Objective C samt C# samt integreras med andra ramverk. Egna enheter samt emulatorer kan användas vid testningen. Unik objektidentifiering stöds likaså datordriven inmatning. (Appium, 2015)

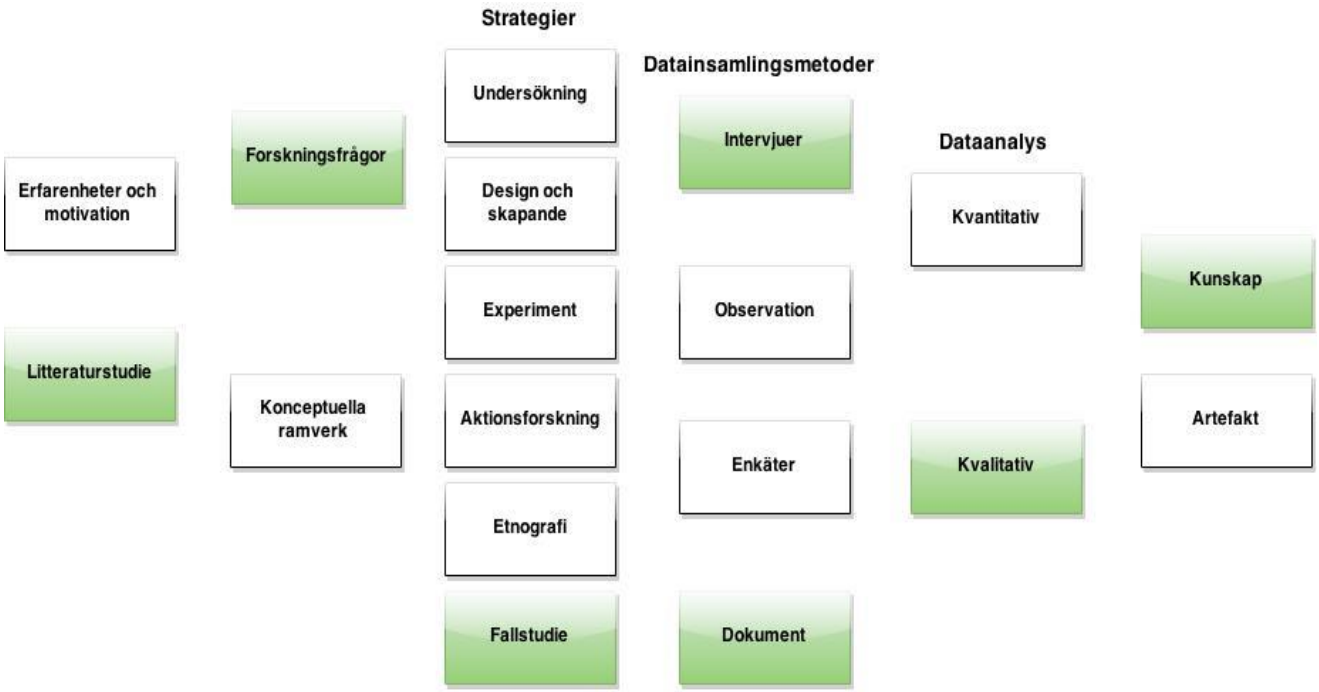
En av de största fördelarna med Appium är att testaren kan utföra tester på samma applikation som användaren sedan kan ladda ner. En av de största nackdelarna är att testskripten inte kan göras på ett flertal enheter samtidigt, vilket innebär att enheterna måste testas en åt gången. (Kipar, 2014)

3 Metod

I kapitlet beskrivs metoder för att samla data och information och tillvägagångssätt för att uppnå arbetets syfte. Som forskningsstrategi har fallstudie valts och de datainsamlingsmetoder vi använt oss av är intervjuer, dokumentstudier samt förstahandserfarenheter. Metodkapitlet beskriver även examensarbetets genomförande samt metodkritik.

3.1 Forskningsprocessen

Figuren nedan beskriver de olika stegen som vi har genomfört under vårt examensarbete. För att få en bättre förståelse började vi med en litteraturstudie som finns beskriven i avsnitt 3.2. Forskningsfrågor formulerades och de finns i avsnitt 1.3. För att besvara forskningsfrågorna genomfördes en fallstudie som finns beskriven i avsnitt 3.3. För att besvara hur testningen av mobila applikationer hos IT-företag utförs idag gjordes intervjuer som finns beskrivna i avsnitt 3.4.1. Frågorna samt transkriberingen av intervjuerna finns i bilagorna 1-7. En dokumentstudie har genomförts för att få en djupare förståelse och den finns beskriven i 3.4.2. Även förstahandserfarenheter beskrivs i avsnitt 3.4.3. Data som insamlats analyserades kvalitativt i avsnitt 3.6.



Figur 3 - Modell över examensarbetet. Källa: Oates (2012) s. 33

3.2 Litteraturstudie

Syftet med en litteraturstudie är att samla och presentera vetenskapligt granskade källor, som ska kunna stödja att ny kunskap skapad genom forskningen alternativt som stöd för resultatet av forskningen. (Oates, 2012).

Vår litteraturstudie har framförallt inneburit sökning efter relevanta artiklar på Internet. Vi har sökt litteratur både via Google.com, Google Scholar samt Summon. De mest använda begreppen vid sökning på Internet har varit: *automated mobile testing, automated testing, testing tools, test automation, testing tools for mobile applications, mobile development, challenges with mobile application testing* samt *mobile application testing.*

Vi har använt mestadels engelska sökord för att få ett bredare resultat vilket gett oss en helhetsbild över testområdet. Enligt Oates (2012) är det ett bra sätt att söka efter litteratur på Internet och snabbt kunna hitta relevant material som är vetenskapligt granskat. Dock påpekar hon att det är viktigt att granska artiklarna noggrant och inte enbart sammanfattningen, då den ibland kan vara missvisande. Vi har granskat artiklarna noggrant för att se att de verkligen bidrar med något till vårt examensarbete. Utifrån artiklarna har även backward search använts, det vill säga att vi granskat artiklarnas referenser för att hitta ytterligare relevanta artiklar. Även forward search har använts, där vi hittat artiklar genom citeringar av artiklar.

Vi har även granskat böcker med relevant innehåll för att få en djupare förståelse för området test för att lättare förstå de olika artiklarna. Testverktyg för mobila applikationer och framförallt automatiserade tester är ett relativt nytt ämne, varför sökning på Internet varit mer aktuell och datumen på artiklar varit viktig. Enligt Björklund & Paulsson (2012) är litteraturstudier tidseffektivt och en billig metod för att ta del av information, vilket är en av dess styrkor. Informationen kan dock vara vinklad och inte alltid heltäckande, vilket är viktigt att ta hänsyn till när litteraturstudien utförs. Vi har haft i åtanke att all material som vi sökt efter inte är vetenskapligt granskat utan skrivet av andra som är insatta i det aktuella ämnet.

Litteraturstudien har vi gjort för att skapa oss en teoretisk grund och en referensram för metoderna vi använt för datainsamlingen. Oates (2012) menar att kunskapen och möjligheten att skapa förståelse kring ett område fås genom litteraturstudien. Den utgör grunden för forskningen och ska bidra till forskningens konceptuella ramverk. Innehållet i litteraturen som används ska granskas för att se om dess innehåll är relevant till forskningen. Vår litteraturstudie har varit en relativt stor del av vårt arbete eftersom den gett oss insikt och en djupare förståelse över området, vilket gjort det möjligt för oss att använda oss av relevanta dokumentstudier samt frågor till intervjuer. Litteraturstudien resulterade i information om testverktygen samt artiklar och tidigare forskning inom området. De mest relevanta för detta arbete valdes ut och presenterades i tidigare forskning (se 1.6) samt bakgrund (se 1.1). Information om testverktygen ledde oss till att vi kunde välja ut fyra testverktyg med hjälp av guiden av Plotytsia som beskrivs i avsnitt 3.5.

3.3 Forskningsstrategi

Inom forskning finns ett flertal olika forskningsstrategier att välja mellan och de har olika för- och nackdelar. De vanligaste sex typerna enligt Oates (2012) är: *undersökning, experiment, design & skapande, aktionsforskning, fallstudie* samt *etnografi.*

Detta arbete utfördes som en fallstudie hos samarbetspartner och vi ville undersöka hur företaget arbetade med testning av mobila applikationer, vilka utmaningar som fanns och hur testningen kunde förbättras av införandet av ett testverktyg. Valet av fall var en unik möjlighet då vi blev tillfrågade då vi blev tillfrågade att utföra detta av samarbetspartner. Oates (2012) menar att en fallstudie är en passande strategi för denna typ av undersökningar. Fallstudien definieras genom en djupgående undersökning av ett eller flera fall i dess verkliga miljö. Vidare menar hon att fallstudien fokuserar på djupet snarare än bredden och forskaren får en helhetsbild över området. Detta genom att använda flera olika datakällor och datainsamlingsmetoder såsom intervjuer, dokument, formulär och direkta observationer. I en fallstudie försöker forskaren få med så mycket som möjligt av de olika relationerna och faktorerna i det studerade fallet, för att få ett bra

sammanhang. Genom det kan en detaljerad bild av fallet skapas och förklaringar till hur och varför saker händer ges. (Oates, 2012)

Explorativa, deskriptiva, explanativa och *normativa* är olika vetenskapliga synsätt på forskningsområdet. Björklund & Paulsson (2012) beskriver att deskriptivt synsätt används när en grundläggande kunskap och förståelse för området finns och målet är att beskriva men inte förklara relationer. Vårt examensarbete har haft det deskriptiva synsättet då vi haft en grundläggande kunskap om testområdet och beskriver hur olika testverktyg för mobila applikationer kan användas i testprocessen.

Oates (2012) beskriver att det finns tre olika tidsperspektiv på en fallstudie: *historisk, nuläges* samt *longitudell*. Tidsperspektivet på vårt examensarbete var en nulägesstudie då vi ville beskriva hur testning av mobila applikationer utförs idag. Oates (2012) beskriver en nulägesstudie som ett fall som studeras under en kort tid, som pågår just nu.

3.4 Datainsamling

Paulsson & Björklund (2012) skriver att litteratursökningar och dokumentstudier ger tillgång till sekundärdata medan intervjuer bidrar med primärdata, vilket innebär information som tagits fram specifikt för studien. Nedan presenteras hur vi fått tillgång till primärdata genom våra intervjuer och förstahandserfarenheter samt sekundärdata genom våra dokumentstudier.

3.4.1 Intervjuer

Oates (2012) beskriver intervjuer som en bra datainsamlingsmetod när forskaren vill komma åt detaljerad information alternativt när det finns öppna eller komplexa frågor. För av besvara den första frågan om hur testningen av mobila applikationer hos IT-företag utförs idag och vilka utmaningar det finns, har vi genomfört intervjuer. Intervjuerna har skett hos sex olika IT-företag för att få en bättre inblick i testprocessen och för att kunna generalisera. Våra intervjufrågor har varit mestadels öppna för att få en djupare information, men även några raka frågor där det enbart krävts ett ”ja/nej” svar. Respondenterna har haft följande befattningar: en testledare, tre testare, tre utvecklare samt en utvecklingschef.

Det finns tre olika typer av intervjuer: *strukturerade, semistrukturerade* samt *ostrukturerade*. Under arbetets gång har vi gjort fyra semistrukturerade intervjuer. Vid strukturerade intervjuer finns en lista med fördefinierade frågor och respondenten har några olika alternativ att besvara frågan med. Semistrukturerade intervjuer har också fördefinierade frågor men respondenten får svara fritt på frågan. Intervjuaren kan även fråga följdfrågor beroende på intervjuns syfte och hur respondentens svar lyder. En ostrukturerad intervju innebär att det inte finns någon lista med frågor utan intervjuaren utgår från respondentens svar och låter denne prata fritt. (Oates, 2012).

Grundfrågorna finns bifogade som en bilaga (se bilaga 1) men har anpassats till respektive företag. Frågorna ställdes för att få en uppfattning om hur testningen av mobila applikationer ser ut idag samt vilka utmaningar det finns. Frågorna kan delas upp i två grupper. Den första gruppen av frågor handlade om vilka plattformar och typer av applikationer som utvecklas samt utmaningarna med utvecklingen. Den andra gruppen handlar om hur testningen av mobila applikationer sker idag, om testverktyg och automatisering samt de största utmaningarna med

testningen. Intervjun hos samarbetspartner handlade också om hur testerna utfördes idag utan testverktyg hos dem vilket gav oss underlag för att besvara den tredje forskningsfrågan.

Intervjuerna har gjorts på plats hos företag som utvecklar mobila applikationer. Intervjuerna utfördes av oss båda och tog ca 45 min. Alla intervjuer har spelats in efter samtycke från respondenterna. Att spela in intervjuer är enligt Oates (2012) ett bra sätt att få med hela intervjun utan att behöva lägga fokus på att anteckna respondenternas svar. Dock ska hänsyn tas till att gestikulering och andra uttryck inte kan återges på inspelningar och att respondenterna kan känna sig nervösa av vetskapen att bli inspelade. Vi har varit medvetna om nackdelarna med inspelningar, men beslutat att fördelarna vägt tyngre istället för att sitta och anteckna under intervjun. Intervjusvaren har transkriberats efteråt som finns med i bilagorna 2-5. Vi har varit medvetna om att vår närvaro kan ha påverkat respondenternas svar. För att minimera vår påverkan har vi försökt vara så neutrala som möjligt och ställt samma frågor som inte varit ledande.

Utöver dessa intervjuer har även mailintervjuer gjorts där vi via mail skickat ut frågor till företag. Det var samma grundfrågor som intervjuerna som gjordes på plats hos företagen, men anpassades sedan till verksamheten. Frågorna skickades till IT-företag som finns längre bort och inte hade möjlighet att träffas för en intervju. Dessa intervjuer gjordes för att komplettera de andra intervjuerna för att få en bättre bild över hur testning av mobila applikationer utförs idag. Mailintervjuerna skickades till tolv IT-företag. Vi fick svar från åtta av dem att de tyvärr inte kunde delta, två svarade inte alls och två inkom med svar på intervjufrågorna. Mailintervjuerna finns transkriberade i bilagorna 6 och 7.

3.4.2 Dokument

Oates (2012) menar att dokumentstudier är ett alternativ som kan användas till intervjuer, observationer och frågeformulär och kan ses som en metod att samla in data. Vi har tittat på dokument hos vår samarbetspartner som rör aktuella system samt tillgång till mobila applikationer som använts i detta examensarbete. Det finns enligt Oates (2012) två olika typer av dokument: hittade och genererade dokument. Med hittade dokument syftas på de dokument som existerar i ett företag, t ex redovisningar, manualer eller rollbeskrivningar. Med genererade dokument syftas på dokument som är producerade i ett visst syfte, t ex forskning. Vi har inte tittat på några dokument genererade specifikt för vårt examensarbete utan fått ta del av företagets interna material. Det interna materialet bestod dels av skapade testfall som vi använde som grund för skapandet av våra testfall (se bilaga 8) men också om hur den manuella testprocessen ser ut idag. Eftersom dokumenten är företagens interna material har vi inte tagit med dessa som bilagor. Dokumentstudierna har hjälpt oss att besvara forskningsfrågorna 2 och 3.

3.4.3 Förstahandserfarenheter

En av examensarbetets metoder har vi valt att kalla för förstahandserfarenheter. Detta eftersom det handlar om våra egna erfarenheter av testningen i de olika testverktygen i det här arbetet. Data som kommit genom det baseras på dessa erfarenheter som vi benämner förstahandserfarenheter. Det har i vårt fall inneburit att installera programvara, skapa och exekvera testfall, felsökning samt dokumentation av testfallen. Data som genererats är direkt ämnat för detta examensarbete och är grunden till analysen i kapitel 6. Anledningen till att denna metod är med i examensarbetet är då erfarenheterna från testverktygen inte kan läsas till genom teori utan måste upplevas och testas. Detta har varit nödvändigt för att uppnå syftet.

3.5 Metod för val av testverktyg

Det finns väldigt många olika testverktyg och alla lämpar sig olika bra beroende på vilka behov som finns för testningen av just en specifik mobilapplikation. För att välja det testverktyg som lämpar sig allra bäst för verksamheten har vi använt oss av denna guide presenterad av Plotytsia (2014) på TestLab4Apps. De första två stegen av guiden användes för att välja ut fyra testverktyg som vi ville fördjupa oss inom och analysera mer detaljerat. Steg tre till elva använde vi oss av för att jämföra de fyra valda testverktygen och analysera dem (se kapitel 6).

De elva stegen är utformade som frågor med en förklaring om vad dessa innebär:

1. **Vilka mobila operativsystem stöds av testverktyget?**
   Testverktyget har givna systemkrav vilket innebär att man måste välja vilket operativsystem (IOS, Android & Windows Phone) och dess olika versioner som fokus ska ligga på.

2. **Vilken typ av mobila applikationer stöds av testverktyget?**
   Vilken typ av applikationer som testverktyget stödjer måste tas hänsyn till. Många testverktyg stödjer inte alla tre typer (Native, Hybrid & Web) av applikationer samtidigt.

3. **Krävs källkoden till testning av den mobila applikationen?**
   Det är inte alltid möjligt att få källkoden för testningen av applikationen. Därför är det viktigt att veta detta vid val av testverktyg då det påverkar hur testningen kommer att utföras. I vissa fall finns lösningar i form av en paketerad app att användas.

4. **Är modifikation nödvändig av applikationens kod?**
   Det är viktigt att veta om applikationen måste förändras för att den ska vara möjlig att testas av testverktyget. Vissa verktyg kräver att tredjeparts bibliotek läggs till i applikationen för att kunna testa den.

5. **På vilket sätt är testskripten skapade?**
   Det finns två olika sätt att skapa testskript där båda har olika fördelar och nackdelar. Det första sättet är att spela in tillvägagångssättet med hjälp av ett färdigt script vilket är det snabbaste sättet. Det andra sättet är att skriva scriptet manuellt vilket inte går lika snabbt men är mer kraftfullt och har fler möjligheter.

6. **Vilket programmeringsspråk används av testverktyget?**
   Det är viktigt att verktygets språk matchar det språk som mobilapplikationen är byggd på eller som företaget använder.

7. **Hur känns specifika objekt igen?**
   Det är bra om det finns en unik objektidentifierare för att minska följderna av förändringar och för att underhålla testskripten. En utvärdering måste göras om testverktyget ger möjlighet att få tillgång till de specifika objekten och hanterade objektbiblioteken.

8. **Stödjer testverktyget datordriven inmatning?**
   Lämpliga drivrutiner ger bättre möjligheter att arbeta med platta filer, kalkylblad och databaslager. Detta är viktigt vid aktivering av testexekvering med olika uppsättningar av data.

9. **Hur detaljerad är testverktygets resultatlogg?**
   För att se resultatet från testningen är det bra att få specifik och detaljerad information i rapporterna såsom misslyckade steg i de olika scenarierna, undantag och andra möjligheter. Att kunna konfigurera rapportens format är också bra.

10. **Vilka möjligheter finns det för integrering med andra verktyg?**
    För att bygga ett bra ramverk med testverktyget så behövs en fungerande integrering med andra verktyg såsom rapporthantering, konfigurationshantering, versionshantering, provhantering etc.

11. **Hur ser testverktygets prissättning ut?**
    Priserna för testverktygen kan skilja sig relativt mycket från varandra, från höga licenskostnader till gratis programvara i form av öppen källkod. Det är därför viktigt att veta att ROI (Return On Investment) täcks och att verktygen följer uppgraderingar. Utöver det är det också viktigt att se vad som ingår i priset som betalas, t ex om support, uppdateringar, molntjänster etc. ingår eller om det blir extrakostnader utöver baspriset.

3.6 Dataanalys

En dataanalys innebär att data som forskaren fått in genom de olika datainsamlingsmetoderna bearbetas och analyseras. Detta för att kunna dra slutsatser och se om det finns några speciella sammanhang eller mönster bland data som insamlats. (Oates, 2012) Vi har analyserat det resultatet vi fått in för att till slut kunna dra slutsatser av det och svara på våra forskningsfrågor.

Vårt examensarbete har genererat kvalitativ data från våra semistrukturerade intervjuer samt dokumentstudier. Enligt Oates (2012) finns det två olika typer av dataanalyser: *kvalitativ* och *kvantitativ*. Med kvalitativ data menas all data som inte är numeriskt och det kan t ex vara textdata, bilder eller ljud. Kvalitativ data används ofta i fallstudier, aktionsforskning eller i etnografisk forskning. Med kvantitativ data menas i sin tur all data som är numerisk och består av siffror och tal och används ofta i olika experiment eller statistiska undersökningar. (Oates, 2012) Kvalitativ data i vårt fall bestod av inspelade ljudfiler, testfall samt testrapporter.

Kvalitativ data från intervjuerna transkriberades först för att få en struktur och tydliga svar på frågorna. Vi letade efter skillnader och likheter och sammanställde sedan dessa för att kunna ge svar på forskningsfrågan om hur testningen av mobila applikationer sker idag.

Jämförelsen mellan de olika testverktygen har presenterats i en egenskaps-tabell (se tabell 2, avsnitt 6.2) som sammanställdes utifrån de 11 stegen presenterade av Plotytsia (2014) samt förstahandserfarenheter från testningen. För att kunna svara på forskningsfrågan på testverktygens olika för- och nackdelar analyserades varje testverktyg utifrån funktionella och icke funktionella krav.

En jämförande analys av hur testning utan och med testverktyg sker har gjorts. Den tredje forskningsfrågan var att se hur införandet av ett testverktyg och dess möjligheter till automatisering kunde förbättra testningen av mobila applikationer jämfört med manuell testning utan testverktyg. För att kunna besvara denna forskningsfråga har vi sammanställt jämförelsen i en tabell (se tabell 3, kapitel 6) för att tydligt se vilka delar som möjliggörs med hjälp av testverktyg. Oates (2012) anser att visuella hjälpmedel är bra för att analysera kvalitativa data.

3.7 Genomförande

Den första delen av arbetet bestod mestadels av litteraturstudier där vi sökte efter information på Internet om testområdet för att skapa oss en referensram. När vi sedan blivit lite mer insatta i området kunde vi försöka få tag på intervjupersoner och skapa intervjufrågor. Intervjufrågorna handlade om hur testningen av mobila applikationer sker idag hos IT-företag för att besvara den första forskningsfrågan. Intervjuerna utfördes relativt tidigt i arbetet och transkriberades sedan.

För att begränsa antalet testverktyg att fördjupa sig i användes enbart välkända testverktyg omnämnda i litteraturen. Testverktygen vi fördjupade oss i var Keynote, AppThwack, MonkeyTalk samt Appium och de hittades under litteraturstudierna. Testverktygen presenteras nedan och de valdes ut med hjälp av de första två stegen från metoden beskriven av Plotytsia (2014).

Vi fick tillgång till tre mobila applikationer från vår samarbetspartner, som vi kunde använda oss av för att få en uppfattning om de olika testverktygen. Huvudsakligen utgick vi från App1 men använde oss även av App2 & App3 som komplement för att kunna utföra testerna. Dessa presenteras nedan:

App1 - Applikationen ingår i ett transportsystem för logistik- och orderhantering. I applikationen kan ordrar hanteras, signeringar göras och det går att både se och ändra status för om en chaufför är tillgänglig eller inte för nya uppdrag. Applikationen är en hybrid-applikation utvecklad i Cordova och den stödjer Android & IOS.

App2 – En native-applikation som stödjer IOS. Applikationen används för underhåll och rapportering som en del i ett system hos verksamheter inom området. Den innehåller funktioner så som möjligheten att rapportera incidenter och kartfunktionalitet.

App3 – En hybrid-applikation utvecklad i Cordova som stödjer både Android och IOS. Applikationen är utvecklad för ett evenemang och är en kommersiell applikation. Den innehåller funktioner som har med evenemanget att göra så som nyhetsflöde, live-uppdateringar samt kartfunktionalitet och vägbeskrivning.

Utifrån detta skapade vi både funktionella och icke-funktionella testfall (se bilaga 8). Som grund för våra testfall som vi skapade, använde vi testfall som vår samarbetspartner skapat i tidigare projekt. Testverktygen laddades ner samt installerades och till molntjänsterna fick vi inloggningar. Med hänsyn till korta licenstider arbetade vi relativt intensivt med testverktygen under en period.

Testverktyget Keynote är ett molnbaserat testverktyg men hade en tillhörande kontorsklient som laddades ner. Testfallen både skapades och kördes i kontorsklienten. Testverktyget AppThwack

är också ett molnbaserat testverktyg och krävde ingen nedladdning av något program till datorn utan gick att köra direkt på hemsidan efter registrering. Testverktyget MonkeyTalk krävde ingen betal-licens för att användas utan det räckte med registrering av medlemskap på hemsidan för att få tillgång till länk där nedladdning av programmet gjordes. Testverktyget kördes sedan rakt från den nedladdade klienten men krävde att programmet Eclipse IDE också skulle vara installerat på datorn. Testverktyget Appium är ett Open Source-verktyg som laddades ned från deras hemsida. För att testverktyget skulle fungera krävdes även installation av Eclipse och några andra program samt inställningar i datorn som skulle göras.

Efter att ha utfört våra testfall och använt testverktygen kunde vi bilda oss en uppfattning om dessa och sammanställde ett resultat. Testverktygens egenskaper jämfördes med varandra och en sammanställning av detta gjordes i form av en tabell som stöd för analys av respektive testverktyg.

Vi har även gjort en jämförelse över testningen utan och med testverktyg som baserades på hur samarbetspartnern testar mobila applikationer idag mot vad testverktygens funktioner kan göra. Detta baseras på intervju och dokument från samarbetspartner och presenteras i tabell 3, kapitel 6. Utifrån analysen har resultatet diskuterats och vi har besvarat forskningsfrågorna samt dragit slutsatser. En reflektion över arbetet fördes och förslag på vidare forskning gavs. Problem som uppkom under arbetet diskuteras också i Reflektioner, avsnitt 7.3.

3.8 Metodkritik

Oates (2012) beskriver att metodtriangulering innebär att två eller flera datainsamlingsmetoder används i syfte att säkerställa validiteten i det som undersöks. För att besvara första forskningsfrågan har intervjuer genomförts. För att besvara andra forskningsfrågan har förstahandserfarenheter använts. För att besvara den tredje forskningsfrågan har vi använts oss av datainsamlingarna intervju, dokumentstudier samt förstahandserfarenheter. Genom våra val av metoder anser vi att vårt examensarbete har gett ett bra underlag för metodtriangulering.

Vårt examensarbete har använt sig av strategin fallstudie eftersom den lämpat sig bäst för vårt syfte som fokuserar mer på djupet om hur testningen av mobila applikationer görs idag, jämförelse av valda testverktyg samt deras för- och nackdelar. Om vi hade haft som syfte att bidra med en guide/modell eller riktlinjer om hur ett testverktyg ska implementeras alternativt användas, skulle strategin ”design & skapande” vara mer lämpad. I vårt fall var syftet att bidra med en jämförelse och beskrivning över testningen idag, vilket var anledningen att vi inte valt den strategin.

Vårt examensarbete har inte varit tillräckligt stort för att rena generaliseringar kan dras om hur testning av mobila applikationer utförs idag. För att kunna göra generaliseringar och öka validiteten skulle vi kunnat utföra fler intervjuer samt haft enkäter som komplement. Däremot har våra intervjuer haft samma grundfrågor och gjorts med IT-företag vilket lett till att vi kunnat dra slutsatser om hur test av mobila applikationer sker idag hos flertal liknande verksamheter. Vi har varit noga med att meddela våra intervju-respondenter om att deltagandet är frivilligt, att intervjuerna spelas in och att vi tar hänsyn till att företagshemligheter. Ur etisk aspekt har vi därför inte heller nämnt företagens namn utan enbart gett en neutral beskrivning av dem.

Tidsaspekten är något som begränsat vårt arbete till antalet intervjuer, testverktyg samt testfall. Om mer tid funnits hade fler typer av applikationer kunna testats i testverktygen för att få en bättre överblick. Likaså hade fler intervjuer kunnat genomföras för att göra generaliseringar. Idealt hade varit att mäta hur testverktygen effektiviserar testningen samt pengarna som kan sparas. Detta har dock inte varit praktiskt möjligt i vårt fall.

Eftersom detta är ett ämne som utvecklas och förändras snabbt, har vi kritiskt granskat de källor vi använt eftersom gamla källor kan bidra med felaktig fakta i vissa fall. Vi har även varit kritiska i urvalet av litteratur för att se att de bidrar till vårt examensarbete. Guiden presenterad av Plotytsia (2014) på TestLab4Apps är inte vetenskaplig utan en blogg-artikel. Dock anser vi att den ändå kan användas eftersom den är skriven av chefen på ett företag inom aktuellt område. Trovärdigheten styrks även genom att artikeln är refererad och använd tidigare av Kipar (2014) i hans studie som jämförde testverktyg för automatisering.

4 Mobila applikationer hos IT-företag idag

I kapitlet presenteras en sammanställning av utförda intervjuer. Dessa intervjuer har gjorts för att uppnå syftet och besvara frågan om hur test av mobila applikationer utförs idag. Intervjuerna är gjorda på plats hos IT-företagen förutom två av intervjuerna som var mailintervjuer. Intervjuerna samt frågorna finns att läsa i bilagorna 1-7.

4.1 Utveckling av mobila applikationer

Samtliga företag utvecklade mobila applikationer förutom företag E som hade fokus på test och krav. Applikationerna utvecklades både till plattformarna IOS och Android. Några av företagen utvecklade även till Windows Phone. Företagen utvecklade både hybrid-applikationer, native-applikationer och vissa utvecklade även mobila webbapplikationer. Hybrid-applikationerna var utvecklade i Xamarin och Cordova.

Under alla intervjuerna framkom det att utvecklingen av mobila applikationer medför många utmaningar. Företag C beskriver utmaningarna såhär: *”Antalet plattformar som ska stödjas och olika enheter, skärmstorlekar samt olika OS versioner. Sedan blir det en utmaning vad man ska stödja om man utvecklar en mobilapplikation vilket leder till att kraven blir viktigare eftersom man där talar om vilka enheter och modeller som stöds.”* (Citat av testledare från företag C, bilaga 4)

Skärmstorlekar, prestanda, olika operativsystem och dess versioner samt olika enheter var de största utmaningarna som alla våra respondenter upplevde sig ha. Företag F menar att fragmenteringen av enheter är en stor utmaning. Ytterligare en utmaning enligt flera av företagen, var valet av enheter som ska stödjas som ska finnas med i kravspecifikationen. Att en kravspecifikation är utförlig och detaljerad var enligt respondenter bland det allra viktigaste.

Främst företag B nämnde vikten av att veta vilken typ av applikation som ska utvecklas, om den kräver uppkoppling eller ska användas offline. Vidare menar företag B att i och med en större mängd information på en liten display, så krävs ett lite annorlunda tänk vid utvecklingen för att presentera data vilket fler av våra respondenter också upplevde som en utmaning. Företag A beskrev AppStore som en extra utmaning då det tar lång tid att få applikationen godkänd och samma process ska gås igenom vid varje uppdatering.

4.2 Test av mobila applikationer

Testerna av de mobila applikationerna utfördes manuellt hos samtliga företag. Utvecklarna själva genomförde tester under utvecklingens gång. Några av företagen påpekade vikten av att försöka minska enhetsberoende fel. Ingen av företagen använde något testverktyg till testerna förutom företag F som använt verktyget TestFlight till några projekt. Några av företagen hade sökt information om olika testverktyg men inte implementerat något och uppgav att lite kunskap fanns om de olika testverktygen. Företag A berättar att *”Vi har kollat på lite olika testverktyg men har inte implementerat något som gör automattester så vi har inget testverktyg som vi använder. En emulator brukar vi dock använda”.* (citat av utvecklingschef, företag A, bilaga 2) För att ett testverktyg ska kunna implementeras så måste man även vara insatt i testverktyget och företag C påpekade att underhåll av automatiseringen kan vara tidskrävande.

I och med att det finns många olika enheter och operativsystem så var alla respondenter av åsikten att automatisering av tester på många olika enheter samtidigt skulle vara bra och spara tid. Likaså var prestandatester, regressionstester och upprepade tester något som skulle vara i behov av automatisering. Företag E menar att regressionstester kring flöden som ofta påverkas är bra att automatisera. Vidare beskriver de att anropen från applikationen till servern skulle kunna automatiseras. Företag B menar också att tester för åtkomst till API:er helst bör automatiseras för att se att API:erna är uppdaterade. Några av företagen påpekade vikten av att få in testningen av mobila applikationer så tidigt som möjligt i utvecklingen.

Acceptanstesterna rekommenderades inte att automatiseras och företag E beskriver att *”Man kan aldrig automatisera en känsla eller en upplevelse”.* (citat av testare, företag E, bilaga 6)

Dokumentationen av testerna gjordes lite olika hos företagen. Företag D beskriver att deras dokumentation av testerna sker i Excel med ja/nej och i bästa fall någon kommentar. Företag A berättar att dokumentationen av testerna är inget som är kvarvarande hos dem. Om det uppkommer buggar i applikationen så skrivs de upp på en tavla som delas av utvecklarna och tas bort när buggen är löst. Företag F beskriver hur de använder Jenkins till enhets- och integrationstesterna. De använder då byggskript som testar och kollar av kod så att ingen kod hamnar på servern som inte genomgått testerna.

5 Test av mobila applikationer med och utan testverktyg

Kapitlet beskriver hur testningen ser ut hos samarbetspartnern idag utan testverktyg baserat på intervju (se bilaga 5). Kapitlet beskriver även förstahandserfarenheter av användandet av de olika testverktygen med hjälp av testfall (se bilaga 8). Informationen om hur testningen ser ut idag baseras på intervjuresultat och dokumentstudier gjorda hos samarbetspartnern. Kapitlet syftar till att redovisa det empiriska resultatet som sedan analyseras för att uppnå syftet.

5.1 Test utan testverktyg

Testningen av App1 sker idag manuellt utan något testverktyg. Applikationen laddas antingen ned på en mobil eller på en emulator. På emulatorn utförs funktionella tester så som att logga in, hantera ordrar och lediglista etc. för att sedan logga ut. Eftersom App1 är en hybrid Cordova-applikation så kan den även testas via webbläsaren vilket också görs idag. För att testa App1 på fysiska enheter laddas APK/IPA ner från en hemsida och installeras sedan i mobilen. Sedan utförs samma funktionella tester som i emulatorn och när dessa är gjorda så avinstalleras applikationen från mobilen. För att se hur App1 fungerar i olika mobiler lånas mobiler av varandra och samma process utförs i de olika mobilerna. Testningen sker som ad-hoc, dvs. utan skapade testfall. Resultatet av testerna dokumenteras i ett Excel-dokument. Denna testprocess är snarlik för App2 och App3.

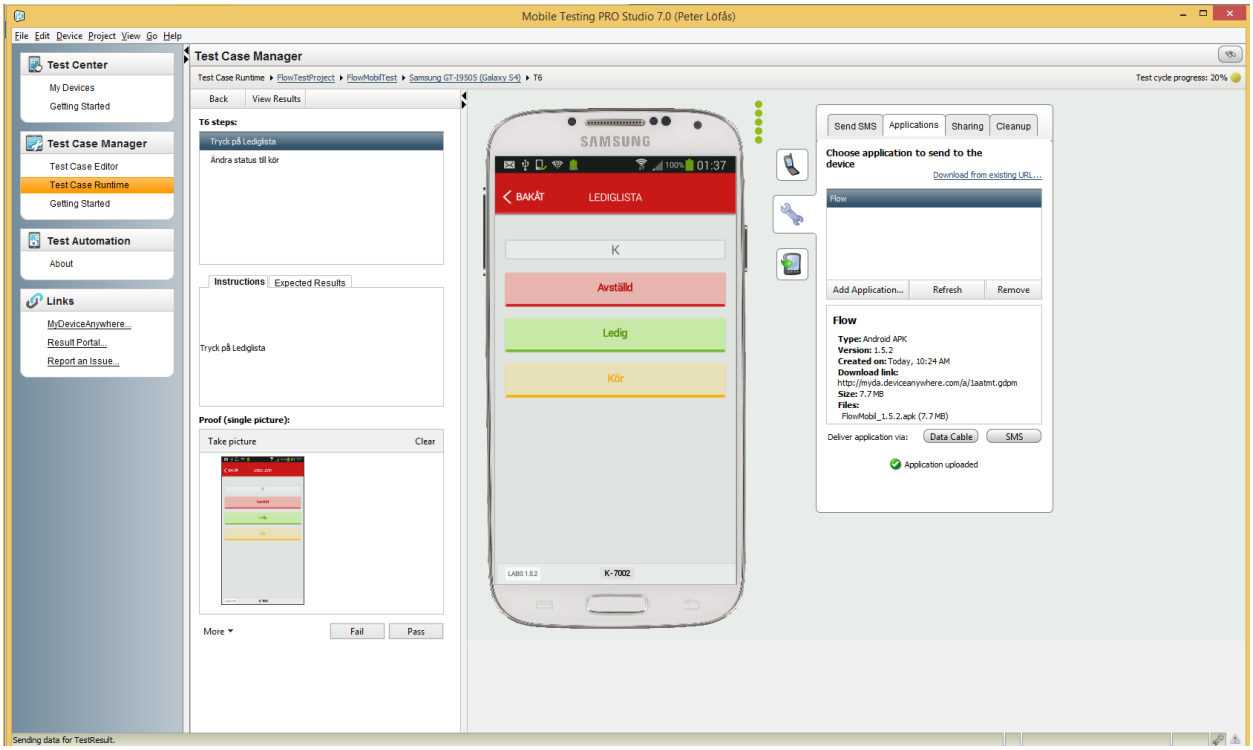
5.2 Test med testverktyg

Nedan presenteras resultatet från användningen av respektive testverktyg. Kompletta testfall finns bifogade i bilaga 8.

5.2.1 Keynote DeviceAnywhere

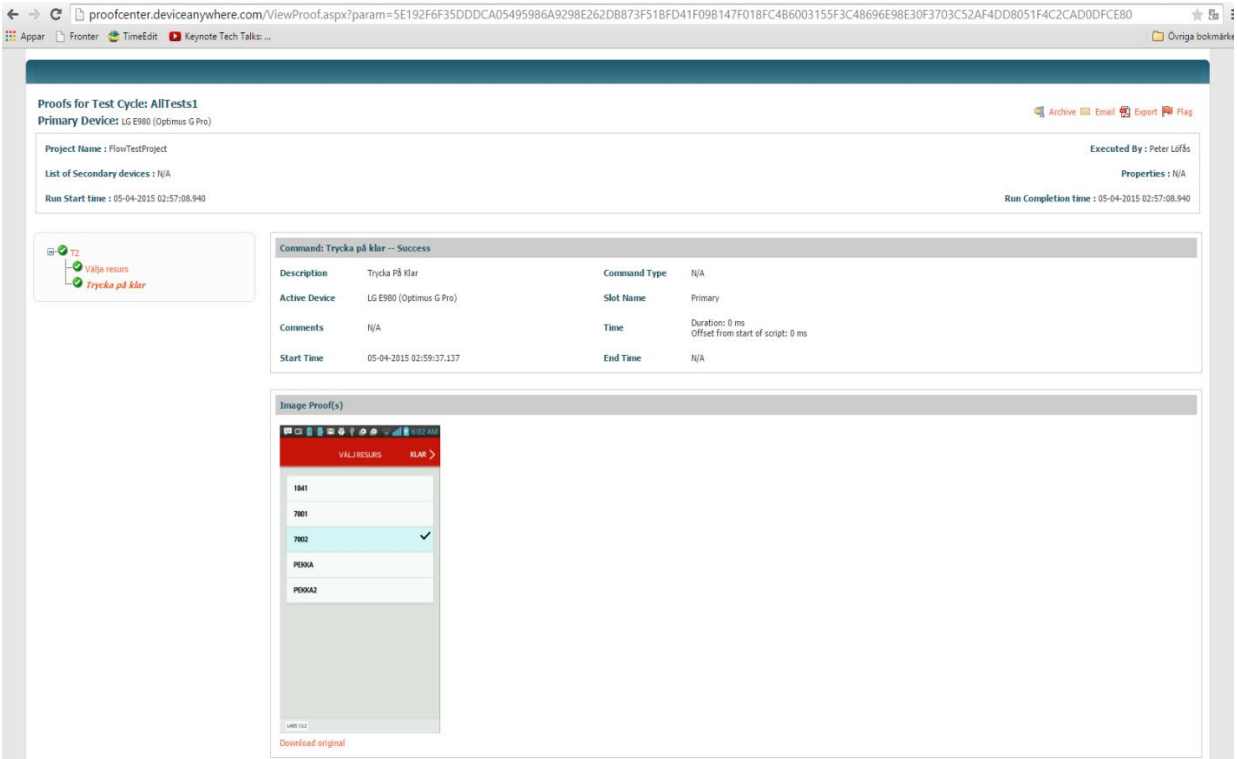
Testverktyget Keynote är en molnbaserad tjänst men krävde att en klient kallad Keynote Studio skulle laddas ned och testningen kördes via den nedladdade klienten. I Keynote Studio kunde ett testfall skapas där de olika teststegen skrevs ner. Ett testprojekt skapades innehållandes testfall. Testfallen lades sedan in i en testcykel och där skulle det även väljas vilka enheter som testet skulle köras på.

Enheterna valdes från en lista inkluderande en kort beskrivning om varje enhet. Det fanns även presenterat om enheten var tillgänglig eller om den var upptagen av någon annan. Testverktyget krävde inte åtkomst till projektet utan en APK/IPA laddades upp på en enhet som valts ut bland de olika tillgängliga enheterna. På det sättet installerades applikationen på enheten och kunde köras på den fysiska enheten via molnet.



Figur 5 - Keynote Studio mitt under en testning baserat på skapat testfall

Testet gjordes sedan utifrån teststegen som tidigare hade blivit uppskrivna i testfallet. Det gick sedan att välja om de olika teststegen skulle dokumenteras genom att spela in video på det eller spara skärmdumpar som bilder. Testet kördes manuellt på en enhet åt gången och man markerade själv om testet blev godkänt eller inte utifrån teststegen som skapats. Detta visas ovan i figur 5.



Figur 6 - Dokumentation av ett testfall med bild bifogad från aktuellt teststeg.

När testet var klart på alla enheter skapades en testrapport som gick att nå via en portal på Internet. Testrapporten visade grön bock/rött kryss beroende på om steget lyckades eller inte och

även detaljer från de olika stegen som utförts samt en översikt (se nedan tabell 1) på de utförda testfallen som tillhörde den specifika testcykeln. Detta visas ovan i figur 6.

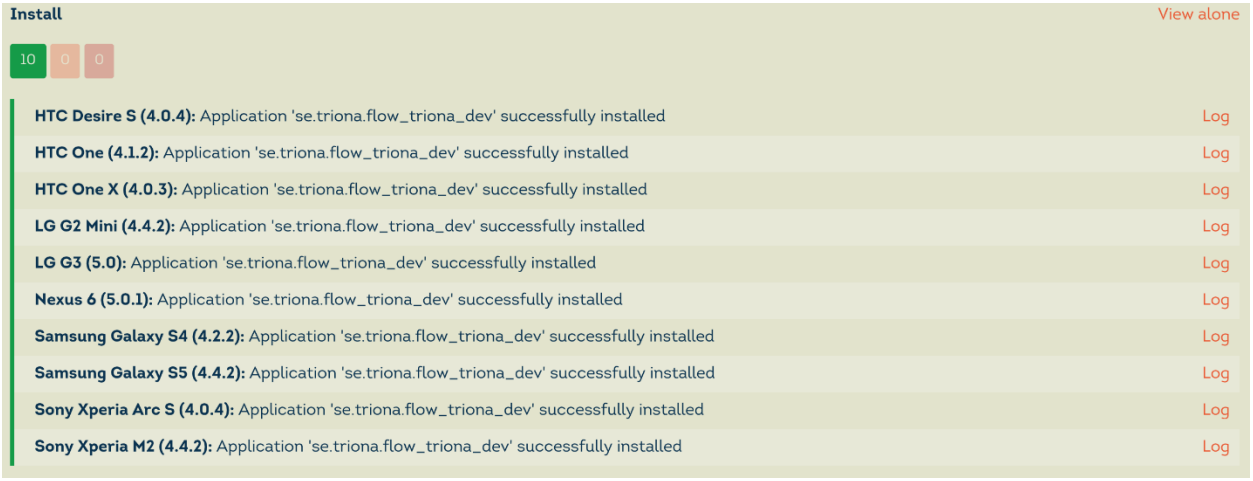
View By Device								
Device	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**
LG E980 (Optimus G Pro)	Success	Success	Failed	Failed	Failed	Success	Failed	Success
Google Nexus 7	Success	Success	Failed	Failed	Failed	Success	Failed	Success
Samsung S4	Success	Success	Failed	Failed	Failed	Success	Failed	Success

Tabell 1 - Tabell på en översikt av testfallsresultat från en testcykel ett testprojekt. Exporterad från Keynote till Excel

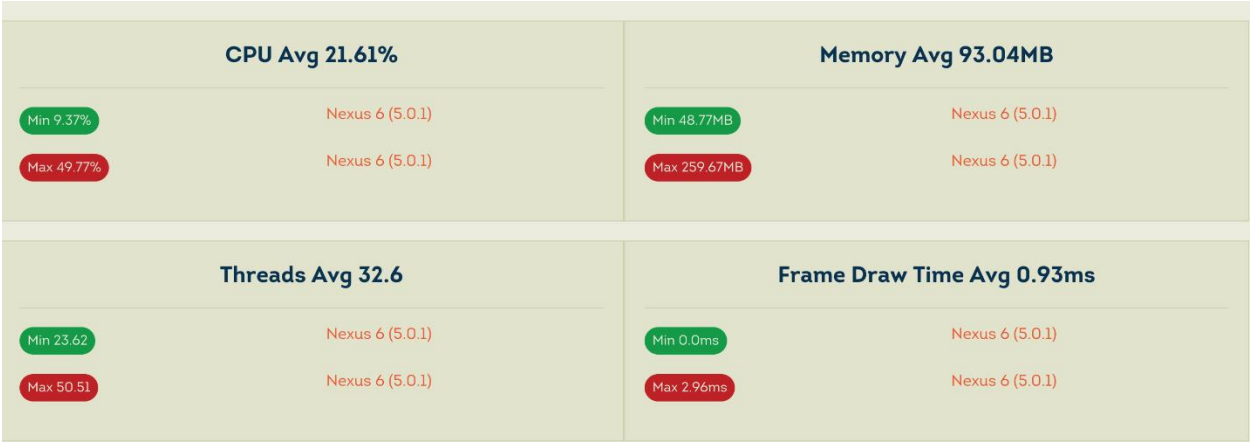
Testfall för App1 gick inte att utföra korrekt eftersom verktyget inte kom åt orderlistan. Detta resulterade i att testfallen T3, T4, T5 och T7 inte gick att utföra och fick därför status ”failed”. (se bilaga 8 för komplett testprojekt inkluderandes testfall). App2 och App3 testades också i Keynote som komplement till App1 eftersom fullt testresultat inte gick att få för den. De testades ad-hoc och resulterade i lyckat testresultat då de funktioner vi testade fungerade.

5.2.2 AppThwack

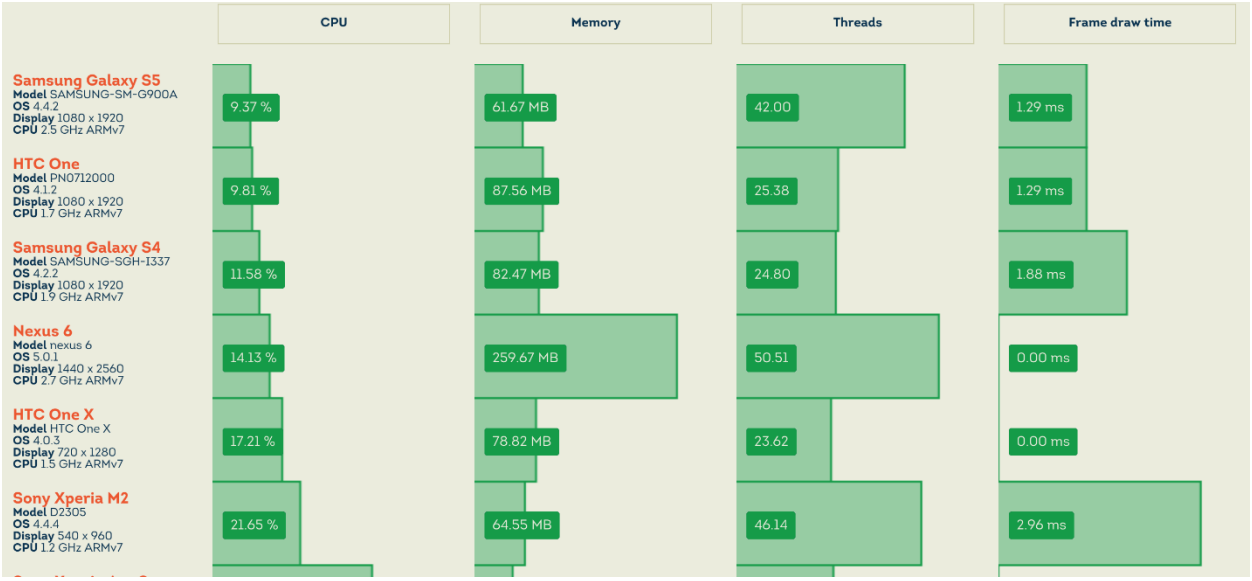
AppThwack är ett molnbaserat testverktyg där ingen klient behövdes ladda ned utan det räckte med registrering och inloggning på hemsidan. Efter inloggning kunde en APK/IPA laddas upp. Det gick sedan att välja vilka enheter som applikationen skulle köras på och sedan antingen ladda upp ett testskript från något annat testverktyg alternativt använda deras egna AppExplorer. AppExplorer är AppThwacks inbyggda program för automatiska funktionella tester av applikationer. App1 kräver en inloggning och AppExplorer stödjer enbart inloggningsfält för native-applikationer så den kunde inte användas för det ändamålet. Dock kunde App1 laddas upp, installeras och avinstalleras samtidigt på valfritt antal enheter som valts ut från listan. AppExplorers testskript sparades inte utan skapades unikt för varje applikation som laddades upp. Både App2 och App3 testades också i AppThwack med liknande resultat som App1.



Figur 7 - Sammanställning av utfört installationstest av App1 på flertal enheter i AppThwack.



Figur 8 - Sammanställning av genomsnittlig prestanda för hur mycket App1 kräver av en viss enhet i AppThwack

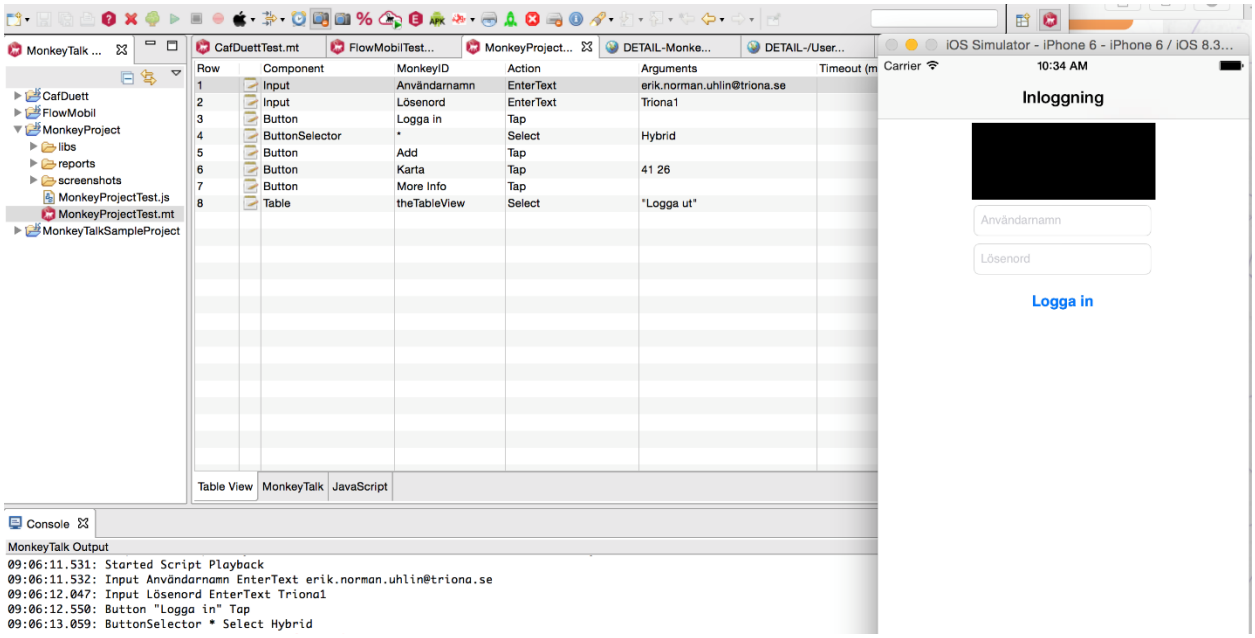


Figur 9 - Översikt på prestanda på flertal enheter i AppThwack

I testverktyget gick det även att se hur mycket prestanda det tagit för en enhet att installera och använda applikationen i de olika enheterna. Detta visas ovan i figur 8 och 9. Testresultatet visades i en rapport med bilder och tabeller/diagram och det gick att exportera rapporten. Figurerna 7, 8 och 9 är alla delar av testresultatet.

5.2.3 MonkeyTalk

Testverktyget är ett OpenSource-verktyg som laddades ned på datorn efter registrering på hemsidan. MonkeyTalk klarade av att spela in det som gjorde på en USB-kopplad Android-enhet som sedan sparades som ett testskript som kunde återspelas. Det gick även att testa via en emulator alternativt en WIFI-kopplad iPhone. MonkeyTalk krävde att Eclipse skulle vara installerat så att deras IDE skulle kunna köras. Testverktyget tillät att skapa testskripten manuellt i deras eget språk MonkeyTalk eller i JavaScript. Detta visas i figur 10 nedan där testskript har spelats in.



Figur 10 - MonkeyTalks IDE med inspelat testskript samt datorns inbygga emulator.

MonkeyTalk spelade dock inte upp testskriptet fullt ut för just App1 som är en hybrid-applikation gjord i Cordova. Uppspelningen av App1 fungerade för textfälten men inte för knapptryckningar så det gick inte att logga in i själva applikationen och genomföra det planerade testfallet (se bilaga 8). För App2 som är en native-applikation så fungerade dock testverktyget fullt ut efter ändring av några ID:n. Testverktyget krävde att ett bibliotek, en så kallad agent, skulle läggas till i projektet för att det skulle kunna köras, vilket kunde göras efter en guide från hemsidan. Därför krävdes tillgång till projektet men däremot inte till någon APK/IPA. Testresultatet presenterades i en HTML-sida där resultatet även kunde exporteras. App3 testades inte i MonkeyTalk då vi inte hade tillgång till projektet.

5.2.4 Appium

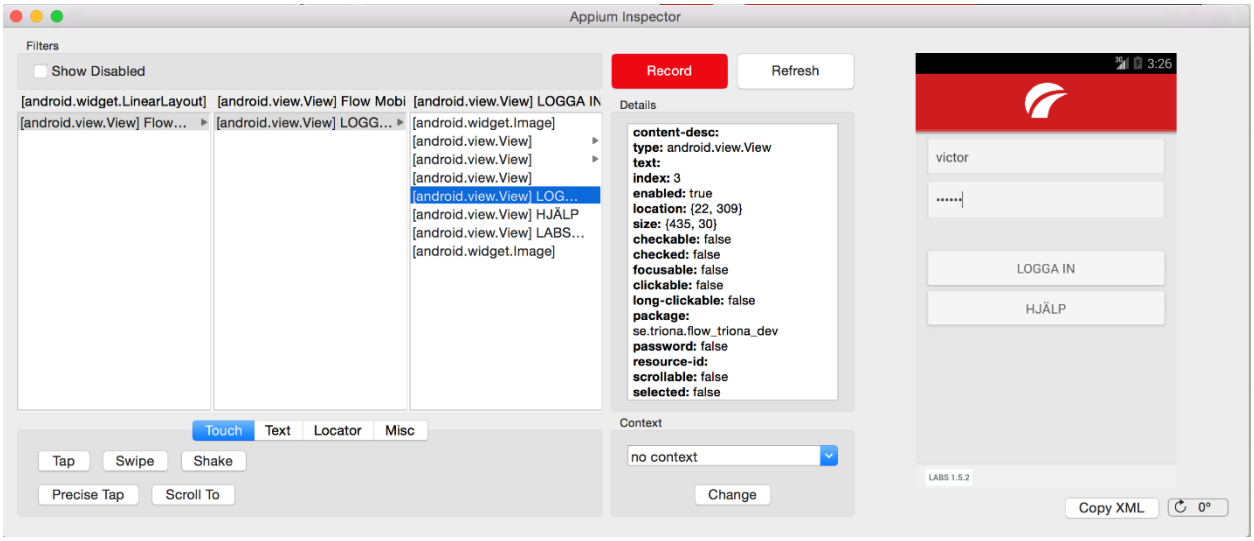
Appium är ett Open Source-verktyg som laddades ner och kördes från datorn. För att testverktyget skulle fungera krävdes även JDK och Android SDK på datorn. Testverktyget kunde integreras med andra verktyg, t ex att testskripten skapades i ett annat verktyg och kördes sedan i Appium Inspector som tillhörde testverktyget. Testskripten kunde skapas i Java, Ruby, Node.js, Python, Objective C samt C#. Ett exempel på skapat testskript visas nedan i figur 11.

```
import io.appium.java_client.AppiumDriver;
import org.openqa.selenium.remote.DesiredCapabilities;
import java.net.URL;

public class {scriptName} {
    public static void main(String[] args) {
        DesiredCapabilities capabilities = new DesiredCapabilities();
        capabilities.setCapability("appium-version", "1.0");
        capabilities.setCapability("platformName", "Android");
        capabilities.setCapability("platformVersion", "4.4");
        capabilities.setCapability("app", "/Users/MacBookPro/Dropbox/EXJOBB/Material fra˚n Triona/FlowMobil_1.5.2.apk");
        wd = new AppiumDriver(new URL("http://0.0.0.0:4723/wd/hub"), capabilities);
        wd.manage().timeouts().implicitlyWait(60, TimeUnit.SECONDS);

        wd.findElement(By.xpath("//android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widget.LinearLayout[1]
        "/android.view.View[1]/android.view.View[1]/android.view.View[1]/android.widget.EditText[1]")).sendKeys("victor");

        wd.findElement(By.xpath("//android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widget.LinearLayout[1]
        "/android.view.View[1]/android.view.View[1]/android.view.View[2]/android.widget.EditText[1]")).sendKeys("123456");

        wd.findElement(By.xpath("//android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widget.LinearLayout[1]
        "/android.view.View[1]/android.view.View[1]/android.view.View[3]")).click();
        wd.close();
    }
}
```

Figur 11 - Appiums testskript skapat i Java. Skriptet visar stegen för inloggningen till App1.

Appium gick att kopplas till en fysisk enhet via USB-kabel. APK/IPA installerades då antingen på den fysiska enheten eller på emulatorn. Testverktyget krävde ingen åtkomst till projektet mer än adressen till var APK/IPA fanns sparad.

Testverktyget spelade dock inte in det som gjordes i App1 på den fysiska enheten eller emulatorn utan man fick lov att manuellt hitta elementen i Appium Inspector för att klicka sig vidare till t ex en knapp för att kunna logga in (se figur 12). Om stegen gjordes manuellt genom att klicka sig vidare i Appium Inspector så fungerade både inspelning av testskript samt att spela upp det i emulatorn eller i fysisk enhet.



Figur 12 - Bild från Appium Inspector, där man klickar sig manuellt till de olika elementen. Till höger i bilden syns emulatorn med inloggningsskärmen på App1.

Testresultatet redovisades i form av en debug- och errorlogg så någon visuell dokumentation gick inte att få fram.

6 Analys av test och testverktyg

Kapitlet beskriver testverktygens egenskaper samt för- och nackdelar. Informationen till detta är dels hämtad från kapitel 2 men även baserad på förstahandserfarenheterna från användningen av de olika testverktygen. Kapitlet syftar till att analysera insamlat data för att ge svar på våra forskningsfrågor och utifrån det dra slutsatser.

6.1 Utmaningar med test av mobila applikationer

Alla intervjuade företag utförde idag sina tester av mobila applikationer manuellt utan testverktyg. De intervjuade företagen är inte ensamma om det utan teorin beskriver också att testningen idag till största delen sker manuellt utan testverktyg. Som vi beskrev i avsnitt 2.2.2 så visade studien av Smartbear (2014) att manuell test fortfarande står för närmare 28 % av utförda tester på mobila applikationer medan automatiska tester stod för drygt 18 %. Dessa 18 % visar dock att det ändå utförs relativt mycket automatiska tester med testverktyg. Även om de intervjuade företagen inte hade implementerat testverktyg så hade många av dem kollat på alternativ och tyckte att det vore bra att ha automatiska tester på t ex prestandatester, regressionstester samt test på access mot API:er.

Utmaningar som framkom under intervjuerna med företagen var huvudsakligen att täcka test på alla olika enheter, operativsystem och versioner av dessa. Det styrks också genom det som sägs i teorin som de största utmaningarna för IT företagen. I bakgrunden (se 1.1) beskrevs hur Mao & Xin (2014) ansåg att de största utmaningarna var de snabba tekniska förändringarna inom mobila enheter, alla olika plattformar, storlekar och enheter som applikationen måste anpassas för samt testningen av dessa.

6.2 Testverktygens för- och nackdelar

Nedan beskrivs testverktygens egenskaper, respektive testverktygs för- och nackdelar samt utmaningar överlag med testverktyg för att besvara forskningsfråga två.

Tabellen nedan beskriver de olika egenskaperna som de utvalda testverktygen har. Kriterierna är baserade dels på guiden från Plotytsia (2014) men även från förstahandserfarenheterna av användningen av de olika testverktygen. Tabellens syfte är att stödja jämförelsen mellan de olika testverktygen för att besvara forskningsfrågan om testverktygens för- och nackdelar.

Kriterier	KeyNote DeviceAnywhere	AppThwack	MonkeyTalk	Appium
Operativsystem som stöds	IOS, Android & Windows	IOS & Android	IOS & Android	IOS, Android & Windows
Mobila applikationer som stöds	Native, Hybrid & Webb	Native, Hybrid & Webb	Native, Hybrid & Webb	Native, Hybrid & Webb
Enhetsstöd	500+ fysiska enheter via molntjänst samt egna enheter	300+ fysiska enheter via molntjänst	Emulator, egna enheter samt enheter via molntjänst	Emulator, egna enheter
Krävs källkoden	Nej	Nej	Nej	Nej
Modifiering av kod	Nej	Nej	Ja	Nej
Hur skapas testskripten	Manuellt eller inspelning	Testskript kan laddas upp från andra verktyg, eller deras egna inbyggda	Manuellt eller inspelning	Manuellt. Inspelning stöds delvis.
Vilket programmeringsspråk används	Java och deras egna inbyggda	-	MonkeyTalk, JavaScript	Java, Ruby, Node.js, Python, Objective C samt C#
Objektidentifiering	Ja	Nej	Ja	Ja
Stöds datordriven inmatning	Ja	Nej	Ja	Ja
Hur detaljerad är resultatloggen	HTML-rapport med fullt testresultat. Möjlighet till export.	Full rapport med testresultat. Möjlighet till export.	HTML-rapport samt XML-rapport. Möjlighet till export.	Resultatlogg i form av debug- och errorlogg
Visuell dokumentation i resultatloggen	Bilder och video	Bilder	Bilder	Nej
Prestandaöverblick	Nej	Ja. Visas med hjälp av diagram och tabeller	Ja	Nej
Integrering med andra verktyg	Ja	Ja	Ja	Ja
Prissättning	Gratisversion eller köpversion 180-750/månad	20-500/månad	Open Source alternativt köpversion	Open Source

Tabell 2 - Beskrivning över testverktygens egenskaper. Informationen är hämtad från avsnitt 2.3.

6.2.1 För- och nackdelar med respektive testverktyg

Keynote
Fördelar med Keynote var utifrån våra egna upplevelser att testverktyget var enkelt att arbeta och komma igång med. Det är en stor fördel att kunna skapa testfall med inkluderande teststeg direkt i verktyget. Ytterligare en fördel är att det inte krävs några egna enheter eftersom testverktyget tillhandahåller ett brett utbud av enheter via molnet. Enheterna har inga begränsningar utan full kontroll fås över dem och installerar själv applikationen på enheten. Testresultatet visades i en komplett testrapport med utförlig information om utfört test. Testrapporten var väldigt bra och beskrivande eftersom den innehöll detaljer om allt som hade gjorts samt även visuell dokumentation. Exempel på detta finns i figur 6.

Nackdelar som vi upplevt med Keynote är att det kan hacka i själva uppkopplingen och det blir fördröjning i interaktionen med enheten. För att använda sig av Keynote måste internetuppkoppling finnas vilket är en nackdel även om vi inte blivit påverkade av det under våra tester. Dessa nackdelar stärks även av Shlykov (2013) som vi beskrivit i avsnitt 2.3.1. Shlykov (2013) beskrev även att det är en nackdel att det blir väldigt dyrt om man vill få tillgång till all funktionalitet. Detta är något som vi också upplevde eftersom vi inte hade tillgång till Enterprise-versionen som även stödjer automatiserade tester och kunde därför inte se hur dessa fungerar. Ett problem som uppstod när vi testade App1 var att funktioner som berörde orderlistan, som hämtas från en server, inte visades i verktyget. Funktionen finns i applikationen och fungerar korrekt i en fysisk mobil men visas inte i testverktygets enheter.

AppThwack
Fördelar var att testverktyget AppThwack var utifrån våra upplevelser väldigt lätt att sätta sig in i eftersom det var lätt att förstå och räckte med registrering på hemsidan samt att inget program behövde laddas ner. Det krävdes heller ingen åtkomst till projektet utan det räckte att ladda upp en APK/IPA, vilket underlättade testarbetet. Testverktyget är väldigt användbart och bra när man vill testa att en applikation kan laddas upp och installeras på flertal enheter, vilket kan göras samtidigt. Den visar även hur mycket prestanda olika applikationer kräver på olika enheter, se figur 8 och 9. Likaså är testverktyget bra när layouten samt funktioner som inte kräver inmatning ska testas på flera enheter.

Nackdelar som vi upplevt med AppThwack var att testverktygets AppExplorer inte klarar av att logga in via webbformulär om det inte är en native-applikation. Detta resulterade i att App1 enbart kunde testas på installation/avinstallation.

MonkeyTalk
Fördelar med MonkeyTalk utifrån våra egna upplevelser är att testverktyget kunde laddas ned gratis på datorn efter enbart registrering på hemsidan. Att skapa och ändra i testskriptet i MonkeyTalk var relativt lätt eftersom deras eget språk var enkelt att förstå. Testskripten kunde skapas genom inspelning. Efter inspelningen kunde testskriptet ändras manuellt rakt i koden för att t ex testa hur det blir när fel användarnamn matas in. Ytterligare en fördel är att det stödjer unik objektidentifiering vilket även Kipar (2014) nämnde som en av testverktygets fördelar (se avsnitt 2.3.3).

Nackdelar som vi upplevde med MonkeyTalk var att testverktyget inte fungerade fullt ut med applikationer utvecklade i Cordova trots att stöd för hybrid-applikationer anges. Detta gjorde att App1 inte kunde testas fullt ut i testverktyget. Eftersom ett bibliotek måste läggas till, krävdes åtkomst till projektet och det räckte inte med att endast ladda upp APK/IPA. För vår del innebar det problem med överföringen av projektet då vi inte hade Cordova installerat på datorerna. Eclipse skulle också vara installerat på datorn för att MonkeyTalks IDE skulle fungera.

Appium
Fördelar med Appium utifrån våra egna upplevelser är att testverktyget är Open Source och kunde laddas ner och installeras utan licenser. Testverktyget stödjer många olika språk för skapande av testskript, vilket gör valmöjligheterna och integrering med andra ramverk större. Appium stödjer även unik objektidentifiering.

Nackdelar med Appium var att installationen var komplicerad och krävde många olika steg samt nedladdning av andra program vilket var för oss en omständlig process. För att få Appium att fungera krävdes även att Android och Java skulle vara installerade och konfigurerade på datorerna. Objektidentifiering fungerar enbart genom att leta på rätt objekt via elementens namn så som vi beskrev i avsnitt 5.2.4. Testverktyget genererade ingen komplett testrapport utan endast en debug/error-logg vilket gjorde att någon överblick över utfört test inte gick att få fram.

6.2.2 Utmaningar med testverktyg
Utöver de funktionella kraven som vi jämfört i tabellen i avsnitt 6.2 så analyserades verktygen också utifrån icke funktionella krav (se bilaga 8). De icke funktionella kraven är baserade på hur de olika testverktygen var att använda samt vad som ansågs i teorin vara som utmaningar med testverktyg.

Precis som studien av Kochhar, Thung, Nagappan, Zimmermann & Lo (2015) beskriven i bakgrunden (se 1.1) visade sig utmaningarna med testverktyg vara ett flertal. Otillräcklig dokumentation och att testverktygen kunde vara svåra att lära sig är något som vi också upplevt. Hur installationen var och hur användarvänliga de olika testverktygen var skiljde sig mycket från varandra. Det påpekades också i intervjuerna att det kräver tid och kompetens för att sätta sig in i testverktygen.

Vid test av App1 i MonkeyTalk stötte vi på problem då hybrid-applikationer utvecklade i Cordova inte verkade stödjas fullt ut trots att testverktyget stödjer hybrid-applikationer. Detta är något som också bekräftas i teorin av utvecklarna i studien gjord av Kochhar, Thung, Nagappan, Zimmermann & Lo (2015). Utvecklarna påpekade att problem kunde uppstå när applikationerna som skulle testas i testverktygen var utvecklade med olika tekniker.

6.3 Jämförelse mellan test utan och med testverktyg

Avsnittet innehåller jämförelse på hur testningen av mobila applikationer sker idag utan och med testverktyg baserat på intervju, dokument, teori samt förstahandserfarenheter. Detta syftar till att svara på vår forskningsfråga om hur testningen av mobila applikationer kan förbättras med införandet av testverktyg samt genom automatisering jämfört med manuell testning utan testverktyg.

Tabellen nedan beskriver hur de olika delarna vid testningen av mobila applikationer ser ut utan testverktyg samt vilka delar som kan täckas med hjälp av de olika testverktygen. I den första kolumnen finns olika delar av testprocessen indelade. I den andra kolumnen presenteras det som är möjligt att göra utan testverktyg. Den tredje kolumnen presenterar det som går att göra med hjälp av testverktyg. Den fjärde kolumnen presenterar vad respektive testverktyg klarar av.

♠ = Keynote
♣ = AppThwack
♥ = MonkeyTalk
♦ = Appium

	Utan testverktyg	Med testverktyg	
Enheter	**Tillgängliga fysiska enheter** **Emulator**	**Tillgängliga fysiska enheter** **Emulator** **Enheter via molnet**	♠ ♥ ♦ ♥ ♦ ♠ ♣
Skapande och utförande av test	**Testfall i Excel/Word** **Manuella funktionella tester**	**Testfall i verktyget** **Manuella funktionella tester** **Automatisering:** **– Spela in egna testskript** **– Skriva egna testskript** **– Importera testskript från andra ramverk** **– Schemalägga automatiserade tester** **– Stöd för test på flera enheter samtidigt** **– Prestandaöversikt**	♠ ♠ ♠ ♣ ♥ ♦ ♠ ♥ ♦ ♠ ♥ ♦ ♣ ♦ ♠ ♠ ♣ ♣ ♥
Dokumentation	**Testresultat i Excel/Word** **Egna skärmdumpar**	**Visuellt resultat** **– Skärmdumpar:** **– Video** **Fullständig testrapport**	♠ ♣ ♥ ♠ ♣ ♥ ♠ ♠ ♣

Tabell 3 - En jämförelse över vilka delar som täcks av de olika testverktygen jämfört med utan testverktyg

Vid test utan något testverktyg får de fysiska enheter som finns tillgängliga användas i kombination med emulatorer. Med testverktyg finns även utöver dessa, möjligheten att använda enheter via molnet. Detta stöds dock enbart av Keynote och AppThwack.

Utan testverktyg så skapas testfallen i Excel alternativt Word och de funktionella testerna görs manuellt med en enhet åt gången. Keynote är det enda testverktyget där testfall kan skapas och manuella funktionella stöds direkt i verktyget. Alla testverktyg stödjer automatiska funktionella tester men dessa kan enbart schemaläggas i Keynote. Förutom AppThwack kan samtliga testverktyg spela in och skapa egna testskript. I AppThwack samt Appium kan även testskripten importeras från andra ramverk. Keynote samt AppThwack stödjer test på flera enheter samtidigt. En överblick på prestandan som aktuell applikation krävt under testet stöds av AppThwack samt MonkeyTalk.

Utan testverktyg får testresultatet skrivas i Excel alternativt Word och man kan bifoga egna skärmdumpar. Förutom Appium så visades testresultatet visuellt av samtliga testverktyg med skärmdumpar från olika teststeg och ev. problem. Keynote stödjer även videoinspelning på utfört test. En fullständig testrapport som också kunde exporteras stöds av Keynote samt AppThwack.

7 Diskussion och slutsats

Kapitlet innehåller svar på forskningsfrågorna samt diskussion kring resultatet. Kapitlet innehåller även resultatkritik samt förslag till vidare forskning. Svaret på första forskningsfrågan är baserad på avsnitt 6.1 i analysen. Andra forskningsfrågan baseras på 6.2 och den tredje forskningsfrågan baseras på 6.3

7.1 Svar på forskningsfrågor

Syftet med examensarbetet är att beskriva hur testning av mobila applikationer sker idag. Ett annat syfte är även att utvärdera hur testverktyg kan användas vid testning av mobila applikationer och jämföra detta mot manuell testning utan testverktyg.

För att besvara syftet ställdes en forskningsfråga inkluderandes delfrågor och dessa besvaras nedan.

Hur kan olika testverktyg användas för tester av mobila applikationer?
Delfrågor:

1. Hur utförs testningen av mobila applikationer idag inom IT-företag och vilka utmaningar finns det?
Utifrån resultatet av intervjuerna sker testningen av mobila applikationer idag mestadels manuellt och utan stöd av testverktyg. Anledningarna att testverktyg inte implementerats var främst att det var mycket att sätta sig in i och krävde både tid och kompetens. Enhetstesterna gjordes under utvecklingen av utvecklarna och de funktionella testerna gjordes främst utan skapade testfall på de mobila enheter som fanns tillgängliga. De främsta utmaningarna vid testningen var att täcka test på alla enheter, skärmstorlekar, operativsystem och versioner av dessa som applikationen skulle fungera i.

2. Vilka för- och nackdelar finns med testverktygen?
Testverktygen har olika egenskaper och därmed olika för- och nackdelar. Detta gör att de passar olika bra beroende på typ av projekt och mobila applikationer. Molntjänsternas största fördel är att det finns tillgång till väldigt många enheter direkt i testverktyget. Keynote är det mest kompletta och heltäckande av testverktygen, men samtidigt det dyraste för att få tillgång till all funktionalitet. AppThwack är testverktyget där det enklast och snabbast går att testa en installation och dess prestandaöverblick, GUI samt vissa funktioner på flertal enheter och visuellt få rapport på detta. Däremot går det inte att skapa egna testskript och AppExplorer som testar funktionerna automatiskt fungerar enbart på native-applikationer. I MonkeyTalk är det enkelt att skapa och förstå testskripten då det egna språket som används är lättförståeligt. Testverktyget kan användas gratis men däremot krävs modifiering av kod samt att Cordova inte stöds fullt ut. Appium är gratis och stödjer många programmeringsspråk. Däremot är det inte användarvänligt, har en komplicerad installation och ger inget visuellt testresultat som de övriga testverktygen gör.

3. Hur kan testningen av de mobila applikationerna förbättras vid införandet av ett testverktyg samt genom automatisering jämfört med manuell testning utan testverktyg?

Vid manuell testning utan testverktyg måste man ha tillgång till enheten och enbart en enhet kan testas åt gången. Testresultatet måste manuellt skrivas i ett dokument som resulterar i en testrapport. Med hjälp av ett testverktyg ges möjlighet att testa på fler enheter genom tillgången till hundratals fysiska enheter via molnet. Detta innebär att företaget inte behöver ha tillgång till alla enheter och installera applikationen i varje enhet vilket effektiviserar testningen. Testverktygen gör att funktionella tester även kan utföras automatiskt genom att samma testskript används till flera tester vilket också effektiviserar testningen. Det finns tester som är svåra att utföra utan testverktyg, exempelvis prestandatester. Med ett testverktyg kan en överblick fås om hur mycket prestanda det krävdes av en enhet under testet av en applikation.

Med hjälp av testverktyg kan både skapandet av testfall och dokumentation av detta ske i testverktyget med resultatet som en komplett testrapport som även visas visuellt. Detta ger mer struktur jämfört med att använda flera olika dokument och ger en tydligare bild över testningen med hjälp av t ex bilder samt video.

7.2 Diskussion kring resultat

1. Hur utförs testningen av mobila applikationer idag inom IT-företag och vilka utmaningar finns det?
Utifrån intervjusvaren om hur testningen av mobila applikationer ser ut idag framkom det tydligt att testverktyg ännu inte implementerats i många verksamheter. Både intervjuerna och litteraturen har visat att testningen av mobila applikationer fortfarande är under utveckling. Mobila applikationer som delar i system hos företag är fortfarande nytt och testningen av dessa prioriteras därför inte lika mycket. Vi tror att testverktygen kommer att vara mer implementerade i verksamheter om några år eftersom det krävs mer mognad. Användningen av mobila applikationer i företag kommer troligtvis fortsätta öka vilket gör att kraven på dessa också ökar och det kommer att resultera i att de måste testas mer än de görs idag.

Utmaningarna med testningen av mobila applikationer som finns idag kommer inte att minska eftersom skärmstorlekarna och modellerna blir fler. Operativsystem släpper nya uppdateringar vilket gör att versioner av dessa ökar och alla de har olika begränsningar som måste tas hänsyn till. Därför kommer utmaningarna bara att bli fler med utvecklingen och därmed också testningen av mobila applikationer. Det kommer att vara ohållbart att testa manuellt utan testverktyg om alla enheter ska stödjas. Testverktyg kommer därför att vara mer eller mindre ett måste vilket diskuteras mer under fråga 3.

2. Vilka för- och nackdelar finns med testverktygen?
Utifrån de applikationer och den funktionalitet i testverktygen som vi fått tillgång till, finns inget testverktyg som är riktigt komplett för både hybrid- och native-applikationer. Testverktygen och testmiljöerna skiljer sig mycket från varandra och för- och nackdelarna blir därför olika beroende på i vilket projekt eller vilken applikation de används i. Bland annat räckte det inte med att enbart ladda ner ett testverktyg och börja använda det utan vissa av testverktygen krävde installationer som var mer avancerade och krävde flera program som tillägg. Även modifiering av projekt som krävdes av ett verktyg var något som påverkade svårighetsgraden och upplevelsen. Dessa detaljer påverkar beroende på vem som ska använda testverktyget. En testare kanske inte behärskar de mer tekniska bitarna på samma sätt som en utvecklare medan testaren är mer insatt i skapandet av testfall som också kan göras i ett testverktyg.

Det är en fördel för testverktyg så som Keynote att de exempelvis erbjuder automatisering som kan schemaläggas. Dock tillhör dessa Enterprise-delen som kostar betydligt mer och vi har därför inte kunnat testa sådan funktionalitet i vårt examensarbete. Det leder till att det både kan ses som en fördel och nackdel för det specifika testverktyget eftersom möjligheten till detta finns men endast under vissa förutsättningar.

Eftersom vi var obekanta med testverktygen och oerfarna testare så kan testverktygen upplevas som svåra att lära sig. Detta nämns även i bakgrunden (se 1.1) där Kipar (2014) beskriver att testverktyg inte rekommenderas vid oerfarna eller tillfälliga testare, knappa tidsresurser eller om testarna inte är bekanta med testverktyget. Under våra intervjuer framkom det också att testverktyg kräver erfarenhet, tid och kompetens. Detta upplevdes även av utvecklarna i studien gjord av Kochhar, Thung, Nagappan, Zimmermann & Lo (2015).

För- och nackdelar kan läsas i litteraturen men vissa saker måste testas i själva testverktygen för att kunna skapa en uppfattning. Vi hade inte enbart med hjälp från teorin fått reda på att Cordova inte stöds fullt ut av MonkeyTalk utan det blev vi medvetna om först under testningen. Enligt teorin ska hybrid-applikationer stödjas. Teorin kan alltså ses som ett stort stöd men det är väldigt viktigt att även själv testa de olika testverktygen för att bli medveten om problem som kan dyka upp.

3. Hur kan testningen av de mobila applikationerna förbättras vid införandet av ett testverktyg samt genom automatisering jämfört med manuell testning utan testverktyg?
Det som uppkom som de främsta utmaningarna med testningen av mobila applikationer idag var just sådant som testverktygen har som syfte att utföra. Att manuellt installera och testa en applikation på en enhet åt gången, som man även måste ha tillgång till är väldigt tidskrävande. Det löses väldigt enkelt med hjälp av ett testverktyg vilket gör att även om testverktygen inte skulle kunna täcka alla delar i testprocessen så kan de ändå ersätta delar enkelt. Testverktygen har funktionalitet där test av dessa delar kan automatiseras vilket effektiviserar testprocessen avsevärt.

Utan testverktyg kan ofta endast ett begränsat antal enheter ges garanti för att de ska stödjas. Genom testverktygen kan många enheter testas vilket vi anser är en av de största fördelarna med införandet av ett testverktyg. Det i sin tur ska finnas med i kravspecifikationen vilket blir en fördel som kan användas även vid upphandlingar.

7.3 Reflektioner

Resultatkritik
Vår kunskap inom testområdet, inte minst testverktyg, har varit väldigt begränsad vilket lett till att det har varit mycket som varit nytt och som tagit tid att lära sig. Ur tidsaspekt har detta medfört att vi inte haft möjlighet att lägga ner lika mycket tid på själva utvärderingen av testverktygen eftersom mycket tid gått åt till bland annat felsökning och att sätta in sig i området både tekniskt och teoretiskt. Vid problem som uppstod tog vi kontakt med Support hos aktuellt testverktyg vilket tog tid. Eftersom vissa av testverktygen krävde installation av Eclipse och ytterligare tillägg så tog det ytterligare tid eftersom vår kunskap om dem var begränsad.

Vi fick inte App1 att fungera fullt ut i MonkeyTalk eftersom det är en hybrid-applikation utvecklad i Cordova. Trots våra felsökningar lyckades vi inte lösa problemen. Däremot kan det även ha att göra med att vi inte har någon tidigare kunskap om plattformen Cordova och om någon annan som är mer kunnig inom detta område skulle utfört detta, kanske de hade kunnat lösa problemet. Detta skulle i sin tur påverkat vår utvärdering av MonkeyTalk.

Testverktyget Keynote kräver licens vilket medfört att vi inte haft tillgång till all funktionalitet även om samarbetspartnern ordnat med betal-version. Detta har påverkat resultatet och därmed även våra slutsatser genom att vi endast kunnat testa manuellt även om testverktyget erbjuder möjligheten att automatisera tester. Vi kan därför se det som en fördel att den möjligheten finns men kan inte uttala oss om hur det fungerar och hur bra automatiseringen är.

Våra muntliga intervjuer gjordes främst med IT-företag här i närområdet. Ingen av dessa hade något testverktyg och heller ingen egen testprocess för just testningen av mobila applikationer utan de gjordes främst av utvecklarna. Vi kan därför inte uttala oss om hur samtliga IT-företag gör och generalisera utan enbart ha detta som utgångspunkt för våra slutsatser.

Examensarbetets bidrag
Vårt bidrag är en beskrivning om hur testningen av mobila applikationer sker idag samt vilka utmaningar som finns. Denna är gjord genom litteratursökningar samt intervjuer med flertal företag inom området.

Vi bidrar även med en jämförelse på fyra utvalda testverktygs egenskaper samt för- och nackdelar. Denna är gjord genom skapande av en teoretisk referensram kring dem samt användning av testverktygen. Genom att använda testverktygen istället för enbart läsa teori om dem så har vi fått en djupare förståelse för hur de fungerar. Utan dessa erfarenheter hade vi inte kunnat dra dessa slutsatser. Detta anser vi ge arbetet en förhöjd trovärdighet eftersom dessa kan bekräftas mot teorin och bidrar med ny kunskap.

Ytterligare bidrag är jämförelsen mellan manuellt test utan och med testverktyg och hur testningen kan förbättras med testverktyg samt automatisering. Denna är gjord genom att insamlat data samlades i en tabell för att påvisa vad som kan göras med hjälp av ett testverktyg och automatisering jämfört med hur det sker idag.

Sammanfattning
Slutligen kan detta examensarbete sammanfattas till att det idag mestadels utförs manuella tester utan testverktyg och att detta är ett område i en utvecklingsfas. Utifrån de applikationer och funktioner i testverktygen som vi fått tillgång till, finns inget heltäckande testverktyg. Däremot har de studerade testverktygen olika egenskaper som kan användas i olika projekt och effektivisera testningen av mobila applikationer

7.4 Förslag på vidare forskning

Ett förslag på vidare forskning är ett experiment på hur mycket effektivare testningen av mobila applikationer blir under ett visst projekt med hjälp av ett testverktyg. Under detta arbete har inte något speciellt fokus lagts på säkerheten i de molnbaserade testverktygen på hur konfidentiell information hanteras. Detta skulle också vara ett förslag på vidare forskning, där testverktygen jämförs utifrån dessa säkerhetsaspekter.

Referenser

Tryckta källor och litteratur

Blomkvist, P. & Hallin, A. (2014). *Metod för teknologer – Examensarbete enligt 4-fastmodellen.* Studentlitteratur. Lund

Björklund, M. & Paulsson, U. (2012). *Seminarieboken.* Studentlitteratur. Lund.

Eriksson, U. (2008) *Test och kvalitetssäkring av IT-system.* Studentlitteratur. Malmö: Holmbergs

Kohl, J. (2013) *Tap into mobile application testing.* Leanpub

Oates, B.J. (2012) *Researching Information Systems and Computing.* SAGE Publications Ltd. London

Artiklar

Alharthi, A. (2014). *Software testing of mobile applications: Techniques and challenges.* In The Fourth International Conference on Digital Information Processing and Communications (ICDIPC2014) (pp. 6-10). The Society of Digital Information and Wireless Communication

Hughes Systique Corporation (2013). *Test Automation Tools for Mobile Applications: A brief survey*. Hämtad 2015-04-07 från: http://www.hsc.com/Portals/0/Uploads/Articles/hsc_whitepaper_mobileTestAutomation_22Feb2013634971268468845610.pdf

IBM (2012), *Native, web or hybrid mobile-app development*. IBM Software

Kipar, D. (2014). *Test automation for mobile hybrid applications: using the example of the BILD App for Android and iOS.* Turku University of Applied Sciences

Kirubakaran, B., & Karthikeyani, V. (2013*). Mobile application testing—Challenges and solution approach through automation*. In Pattern Recognition, Informatics and Mobile Engineering (PRIME), 2013 International Conference on (pp. 79-84). IEEE

Kochhar S. P., Thung F., Nagappan N., Zimmermann T. & Lo D. (2015). *Understanding the Test Automation Culture of App Developers.* Singapore Management University, Microsoft Research

Madhushani, B. A. L., De Silva, P. H. A. M., Madushanka, W. A. L., Malalagama, M. G. T. H., & Manawadu, D. (2014) *Challenges in Mobile Application Testing: Sri Lankan Perspective.* Compusoft, 3(10), 179-185.

Mao, X., & Xin, J. (2014). *Developing Cross-platform Mobile and Web Apps*. CIGR Proceedings, 1(1).

Shlykov, K. (2013). *Tools for mobile application testing: choices and features review.* Hämtad 2015-05-05 från: http://www.enterra-inc.com/techzone/tools-for-mobile-application-testing-choices-and-features-review/

Övriga elektroniska källor

Appium (2015) *About Appium.* Hämtad 2015-04-17 från: http://appium.io/slate/en/master/?ruby#about-appium

AppThwack (2015) *FAQ.* Hämtad 2015-05-11 från: https://appthwack.com/docs/faq

AppThwack (2015) *Overview.* Hämtad 2015-05-11 från: https://appthwack.com/overview

AppThwack (2015) *Pricing.* Hämtad 2015-05-11 från: https://appthwack.com/pricing

CloudMonkeyMobile (2015) *MonkeyTalk.* Hämtad 2015-04-20 från: https://www.cloudmonkeymobile.com/monkeytalk

Jenkins (2015). *Meet Jenkins.* Hämtad 2015-05-22 från: https://wiki.jenkins-ci.org/display/JENKINS/Meet+Jenkins

Keynote (2015). *Mobile Testing.* Hämtad 2015-05-05 från: https://www.keynote.com/solutions/testing/mobile-testing

Plotytsia, S. (2014). *How to choose the right mobile test automation tool.* TestLab4Apps. Hämtad 2015-04-07 från: http://www.testlab4apps.com/how-to-choose-the-right-mobile-test-automation-tool/

Smartbear. (2014). *What is Mobile Testing?* Hämtad 2015-05-08 från: http://smartbear.com/all-resources/articles/what-is-mobile-testing/

Wikipedia (2015). *API, APK-fil, Cordova, Cross-Platform, Emulator, IDE, IPA-fil, Jailbreaking, JDK, Molntjänst, Open Source, SDK.* Hämtad 2015-05-20 från: http://wikipedia.org

Bilagor

1 Grundfrågor till intervjuer

1. Berätta kort om ert företag!

2. Vad har ni huvudfokus på i ert företag? Har ni utvecklat mycket mobila applikationer?

3. Vilken plattform utvecklar ni till?

4. Vilken typ av applikationer utvecklar ni? (Native-, Hybrid- eller mobila webbapplikationer)

5. Vilka är de största utmaningarna med utveckling av mobila applikationer?

6. Har ni något speciellt testverktyg som ni använder?
   Om inte, varför inte och vet ni något om de olika testverktygen som finns idag?
   Om ja, till vilken typ av tester använder ni testverktyget?

7. Gör ni testerna manuellt eller automatiserat? Om ni testar allting manuellt, på vilket sätt tror ni att automatiserad testning skulle kunna vara till en fördel?

8. Vilka är de största utmaningarna med testning av mobila applikationer?

9. Vilken av delarna tror ni är i störst behov av att automatiseras? Vilken del tror ni är svårast att automatisera?

10. På vilket sätt dokumenteras testningen?

2 Intervju Utvecklingschef, Företag A

Berätta kort om ert företag!
Vi är ett IT-företag som håller på med fastighetssystem för att hantera systemet som stöd för fastighetsägaren. Vi finns på flertal orter i Sverige. Totalt har vi ca 50 anställda varav 35 utvecklare.

Vad har ni huvudfokus på i ert företag?
Majoriteten av kunderna är kommunala bolag. Vi har ett fastighetssystem som är kärnan på allt och det är en stor databas och ett system där flera bolag hanteras. Med mobilappen, som tillhör systemet och tar hand om teknisk förvaltning, kan bilder tas på plats vid felanmälningar eller besiktningar och även scannas den mot streckkod på den fysiska varan.
Utöver det har vi även en portal som kan användas till tredje part, som t ex om en målar-firma ska komma och måla till lägenheten. Portalen är en responsiv webb som de kan använda och ta del av informationen om sina ordrar, skriva ut och avrapportera. Utöver det har vi även ett marknadssystem som sköter allt som har med uthyrningen att göra. Den tar hand om köer, poäng och sådant som ska hanteras när en lägenhet blir ledig och hamnar bland de som kan hyras ut och ska publiceras på nätet. Sedan har vi även ytterligare ett system för fastighetsägarna där man kan läsa in allt om förbrukningar som el och vatten i lägenheten och där kan man se energiförbrukningen i olika grafer.

Har ni utvecklat mycket mobilaa pplikationer?
Ja. Vi har appar som tillhör systemet. Apparna som används för den tekniska förvaltningen är rena appar som är gjorda i Cordova. Entreprenörsportalen är dock ingen app utan är enbart en responsiv webb.

Vilken plattform utvecklar ni till?
Apparna utvecklas både till IOS och Android samt Windows Phone.

Vilken typ av applikationer utvecklar ni? (Native, Hybrid eller mobila webbapplikationer)
Native. Fastighetssystemets app är dock en hybrid-app gjord i Cordova.

Designar ni applikationerna också?
Ja. Det är väldigt många kunder som kör appen och en egen app distribueras till varje bolag. Apparna designas med färger och logga så att det blir det egna företagets grafiska profil och inte vår. Vi har en grunddesign och sen lägger vi på förändringen för respektive företag/kund.

Vilka är de största utmaningarna med utvecklingen av mobila applikationer? T ex jämfört med system.
En av utmaningarna är att det är en väldigt liten yta som man vill göra mycket på. En annan utmaning är alla versioner av operativsystem och telefoner eftersom det kan ge olika typer av buggar. En av våra största utmaningar med det här är att mycket av vår kod ligger otypat i JavaScript vilket är svårt att testa först innan den slutgiltiga versionen som kunden har. Ett av de största problemen har att göra med AppStore eftersom synkningen där inte fungerar på samma sätt för där måste man vänta på Apple för godkännande av nya versioner.

Har ni begränsat er till att stödja ett visst antal modeller av mobiler?

I avtalet står det att vi i botten bara har stöd för två versioner bakom av alla system. Vi har försökt skriva att vi stödjer de senaste versionerna i de olika operativsystemen men inte vilka enheter det begränsas till. Vi har inte behövt det då det inte varit några stora problem utan sådant som gått att lösa.

Hur ser er testprocess ut när det gäller fastighetssystemet?
Vi har egna testare och testningen är lite uppdelad. Gällande fastighetssystemet har vi två team där ena teamet jobbar med teknisk förvaltning och det andra teamet med ekonomisk förvaltning och de jobbar i egna SCRUM-team. Tyvärr har vi inte haft testresurser för att täcka båda teamen så det ena teamet gör testerna själv som planeras i slutet av sprinten. Det testas kontinuerligt av testare så fort en del av utvecklingen är klar. Det andra teamet är tyngre och har planerade tester. De flesta tester som man vill göra är integrationstester men miljön är komplex och vår databas klarade inte av det så vi har gått tillbaka till att köra enbart enhetstester. Dock har vi funderingar på hur vi ska kunna återställa databasen och kunna rulla tillbaka, men databasen är stor och kräver mycket.

Under utvecklingen så testar vi kontinuerligt och sedan fryser vi koden och enbart testar i en vecka. Sedan rättas buggar och en vecka efter kodfrysning går vi in i acceptanstestmiljö hos kunden. Den ligger då där en månad och kunden i sin tur testar den i de saker som kommer med den versionen och sen blir det rättningar av det innan den släpps ut. Vi släpper en ny version tre gånger av året så testprocessen upprepas i samband med detta. Test tar mycket tid men det är svårt att räkna på det innan. Vissa versioner skiljer sig inte lika mycket från förra som en annan version och då kan det bli mer eller mindre test beroende på uppdatering.
På kravspec-nivån kommer test redan före utvecklingen och för att test ska lyckas så ska det komma in så tidigt som möjligt och är med även på Daily Scrum. Så test är hos oss en kontinuerlig process. Det svåraste är att dela upp antal timmar rätt. Beroende på hur duktiga utvecklarna är på att testa sin egen kod, kan de även få olika uppgifter för att få till bra tester.

Har ni något speciellt testverktyg som ni använder?
Vi har kollat på lite olika testverktyg men har inte implementerat något som gör automattester så vi har inget testverktyg som vi använder. En emulator brukar vi dock använda.

Gör ni testerna manuellt eller automatiserat? På vilket sätt tror ni att automatiserad testning skulle kunna vara till en fördel?
Manuellt. Vi har försökt jobba längre ner med APIerna etc. så att man minskar antalet enhetsberoende fel.
Den tekniska förvaltningen, mobilappen, testas av produktägaren (acceptanstest). Han skriver kravspecifikationen och ger infon från början och sen när den kommit ut från utvecklingen så testar han den för att se om det blev så som han själv ville. Sådant kan man inte automatisera eftersom det handlar om hans åsikter och frågor om t ex var man placerat olika knappar och varför.
Däremot så skulle automatiseringen kunna vara bra för just regressionstester.

Vilka är de största utmaningarna med testning av mobila applikationer?
Största utmaningen tror jag är att vi i just vårt fall har flera styrfiler till olika installationer som kan ge upphov till olika saker för trots att de har samma kodbas så har de lite olika inställningar. Ibland kan även utvecklarna glömma att lägga in några referenser.

Vilken del tror du skulle vara bäst att automatisera? Vilken del tror du är svårast att automatisera?
Baksidan. Test av tjänstelagret skulle man tjäna mycket på att automatisera. Att gränssnittet högst upp fungerar som den ska behöver man ändå köra igenom så där tjänar man inte lika mycket på att automatisera, inte i vårt fall. Om det blir upprepningar på samma saker så vore det bra med automatiserat. Prestandatestet skulle man också kunna automatisera vilket skulle kunna vara bra. Eftersom databasen är så stor så kan det ta väldigt lång tid om man skulle råka skriva en felaktig select-sats.

På vilket sätt dokumenteras testningen?
Testerna som körs i sprintarna dokumenteras på boarden med alla tasks. Buggarna som hittas åker in där och den utvecklare som har tid och kan, tar på sig det. Det är dock ingen dokumentation som lever efter utan ärendet stängs först när det är klart. Sedan när vi går in i releasen så ser vi på taskboarden vilka buggar som finns kvar att lösa och då kan vi meddela att vi vet om dessa men kommer att åtgärda dem.

3 Intervju Systemutvecklare, Företag B

Berätta kort om ert företag!
Vi är ett konsultbolag som funnits en längre tid. Vi finns på ett flertal olika ställen både i Sverige och fler länder. Vi har alltid försökt skapa arbetsplatsen där man vill arbeta och vill företaget vill få sina anställda att tycka att det är roligt att arbeta och hittar på roliga saker.

Vad har ni huvudfokus på i ert företag? Har ni utvecklat mycket mobila applikationer?
Företaget har varit inriktat mest på R & D tidigare men numera även mycket mobila applikationer. Huvudfokus i företaget har börjat bli mer och mer på mjukvaruutvecklingen även om den andra delen fortfarande finns kvar och man samarbetar med större industrier.

Utvecklar ni till en specifik plattform?
Företaget utvecklar till både IOS och Android. Jag utvecklar till Android.

Vilken typ av applikationer utvecklar ni? (Native, Hybrid eller mobila webbapplikationer)
Jag har kollat lite på Hybrid-appar men utvecklar Native-appar. Beroende på vad som ska göras så kan det vara smidigare med native eftersom man har mer hårdvarustöd direkt i dem. Nu finns det ju så att man kan göra webb-appar för att kunna ha databaslagring och offline-stöd etc. men det är fortfarande ganska nytt men det finns där så att det går att bygga.

Det verkar vara väldigt många som utvecklar just native-appar?
Ja, det är väldigt vanligt fortfarande att man utvecklar native-appar. Det finns många olika bra lösningar för hybrid-appar också, bland annat Cordova-plattformen (tidigare Phone Gap) med Xamarin. Xamarin kostar en hel del så där ska man komma upp i ett antal plattformar för att det ska löna sig.

Vilka är de största utmaningarna med utveckling av mobila applikationer?
På hybrid-appar i IOS måste sista touchen på designen alltid göras på en MAC så designen görs separat vilket kan vara en nackdel men sen finns det ju många andra fördelar. Största utmaningen kan vara att man behöva en bra kravspecifikation för att man ska veta vad som ska göra.

Det är även ett lite annat tänk man ska ha med just mobila applikationer jämfört med när man utvecklar vanliga applikationer så har man tillgång till så mycket mer prestanda etc. och inget man behöver fundera på. I en mobilapp så ska data hämtas i portioner, datat ska lagras kvar i telefonen för att man ska slippa hämta det igen om man bara gjort små förändringar. Det kan också vara lite svårt att veta ibland när kunden begär en viss funktionalitet att varför de begär just den funktionaliteten. Ofta kan det vara för att de haft den funktionaliteten i den gamla och vill då ha den i den nya också, även om det skulle kunna göras på ett bättre sätt. Det skiljer sig inte jättemycket men man får tänka på att det är en liten enhet och man ska kunna se datat på ett bra sätt. Man måste även tänka på vad appen ska användas för. T ex så om den ska användas på ett ställe utan mottagning så måste alltid vara synkat eller datat nerhämtat i portioner genom t ex EDGE/GSM istället för 3G. Ska man utveckla en app som används i en stad där du kan beställa mat så har man alltid mottagning, vilket inte behöver hantera just det på samma sätt även om man måste ha felhantering med det också.

Designar ni apparna också?
Ja, det gör vi.

Testar ni något själva under utvecklingen?
Själva funktionerna i förra appen som jag gjorde testade jag under utvecklingens gång. När jag testkör så har jag på alla loggar så ser jag där om det dyker upp något som jag kan hantera med en felhantering så att det inte kraschar. Sen hade de själva ett par stycken som var väldigt insatta i utveckling så de har testat den själva också. I den senaste appen hade vi vissa krav om enbart porträttläge på appen och vilken var den senaste versionen av Android som skulle garanteras (Push och liknande skulle då ha fullt stöd) att den fungerar på. De hade enbart krav på en storlek då de hade en platta som den skulle användas på.

Hur har ni gjort när ni testat på flera olika modeller/storlekar?
Nu har inte vi stött på just det med t ex den senaste appen men alla har inte samma telefoner här på bolaget så vi lånar runt och testar den i de olika telefonerna. Vi har inte använt oss av någon simulator eller molntjänster.

Har ni något speciellt testverktyg som ni använder?
Nej, det har vi inte.

Har ni någon koll på testverktyg som finns att tillgå?
Jag har inte kollat något på det när det gäller just mobilutveckling utan då har det varit mer vanlig utveckling och kollat på automatiserade tester på just det. Det finns säkert många olika verktyg som kan implementeras men allting beroende på hur stor applikationen blir. På ett tidigare bolag som jag var på körde vi med en del olika verktyg.

Hur och när tycker du att test ska införas i utvecklingsprocessen?
Allting är beroende på vad för typ av app det är man utvecklar. Är det något som är väldigt kritiskt så har man kanske en mer testdriven utveckling och inför det väldigt tidigt. Det gäller att eliminera så mycket felkällor som möjligt under vägen medan testet är till för att se hur användaren gör. De som är bra på att testa, testar ju hur man inte ska göra för att se vad som händer då. User Input är ju den största delen som måste hanteras. T ex att se vad kan dom göra som inte ska kunna göras egentligen och vad händer då?

Vilken del i mobilutvecklingen tycker du skulle vara mest värd att automatiseras?
Vissa delar är ju svårare att automatisera. Det man alltid ska lägga en automat-test på är att om du har en access mot ett API att kolla att API:et alltid ser likadant ut och inte förändras av någon anledning. Man skulle kunna ha sådana tester även om man skulle gå ut mot ett extern API för då får man reda på om något förändrats i deras API. Det kan t ex vara så att de gjort en förändring i sitt API som gör att applikationen inte fungerar för det kan vara ganska vanligt. T ex om något namn blivit ändrat eller hur strukturen ser ut om det är en hård kontroll på om det är JSON eller XML. Om strukturen förändras så slutar det ju fungera och då sitter man och undrar varför inte mobilapplikationen fungerar längre. Allting beror så mycket på vad det är för app. Om det t ex är något med beräkningar så kan man ju lägga automatiserade tester som kollar att beräkningarna är rätt genom att skicka in ett visst data som du vet ska bli ett visst resultat, så vet man om något förändrats på vägen.

På vilket sätt dokumenteras testningen?
När vi testat under utvecklingen så har vi inte dokumenterat det något utan när det är samma person som utvecklat och testat så åtgärdas felet direkt istället. Annars dokumenteras det för att någon annan ska kunna fixa det och det har gjorts med Excel med en backlog på det.

4 Intervju Testledare & Testare
Företag C

Berätta kort om ert företag!
Vi är ett stort IT-konsultföretag som finns i 15 länder. I Sverige finns drygt 1000 medarbetare och vi finns på ett flertal orter. Vi erbjuder lösningar inom Test, BI, Digitala Kanaler, Cloud & Cyber Security etc.

Vad har ni för huvudfokus?
Vi ska verka lokalt, med fokus på systemutveckling, testledning och projektledning. Vi är ganska stora på Microsoft SharePoint just på det här kontoret.

Vilken teknik används vid utvecklande av mobila applikationer? Till vilken plattform utvecklar ni?
Vi använder Xamarin för att utveckla appar. En av fördelarna med Xamarin är att man kan utveckla apparna till alla olika plattformar (IOS, Android och Windows) och dela på kodbasen. Det blir en typ av hybrid app.

Har ni något speciellt testverktyg som ni använder? Kan du berätta lite mer om det?
Ja, vi använder Microsoft Test Manager. Detta verktyg har dels en separat klient som man kan testa med, det finns även en webaccess som man kan använda. Det har funnits sedan 2008 och det kändes då mera som att det var utvecklat av utvecklare som inte kunde så mycket om test. Men det blir bättre och bättre. Det finns även andra verktyg på marknaden som kan vara bättre, men i och med licenser och annat så använder vi det. Vi sitter ju även i Microsoft miljö när vi utvecklar så är det ger ett mervärde till verktyget, det fungerar även ihop med TFS (Microsoft Team Foundation server) på ett smidigt sätt.

Men fungerar det om man inte sitter i en Microsoft miljö?
Jo, jag har använt felhanteringsdelen för att organisera och vid rättning av buggar. Detta när jag varit ute hos kund eftersom de köpt in ett system som måste testas. Det kan även användas vid acceptanstest då man kan skriva testfall som sedan kan skickas ut till de som ska testa.

Gör ni testerna manuellt eller automatiserat?
Jag använder inte automatiserade tester för jag känner att det oftast inte finns tid till det. Man måste dessutom ha något stabilt som redan fungerar om man ska genomföra automatiserade tester. Ska man automatisera tester så måste man nästan vara med redan från start av ett projekt för att det ska fungera. Det krävs väldigt mycket tid för att genomföra automatiserade tester.

Har ni testat någon mobilapplikation? Användes ni något testverktyg då?
Ja en mobil webb-applikation har testats, men inte med något testverktyg.

Hur gick det till att testa den?
Jag använde en Ipad eller androidplatta för att testa applikationen. Data skickas sedan in till huvudsystemet. När kontrollerna ute på fältet görs så används en applikation i en surfplatta till detta. Då testar jag att alla funktioner fungerar och fel hanteras på korrekt sätt.

Hur dokumenteras testningen?
Det sker i TFS.

Vad är den största utmaningen med test?
En utmaning är att mognaden i företaget angående test är väldigt olika. Sedan så är dåliga krav, icke uppdaterade krav än stor utmaning. Eftersom det jobbas väldigt mycket agilt idag så kan kraven bli lite slarviga i och med att dessa skrivs som user stories, och det kan vara svårt att formulera kraven då. Viktigt med test är att man börjar tidigt i projektet, redan när man börjar prototypa.

Vilka tester genomför utvecklare?
Enhetstester förutsätter jag att de gör.

Vilken del skulle vara mest i behov av automatisering inom test? Utmaningar kring automatiserade tester.
Har man ett system eller webapplikation som är stabilt och inte förändras och kommunicerar med andra system, då kan automatisering vara bra. Det är nog på väg mot att bli mer automatiserade tester men mognaden är nog för låg idag på det och även när det gäller agila metoder.
Automatiserade tester ska även underhållas och där blir det nog ett problem, det kan leda till att man gör det lite för lätt för sig.

Man genomför endast testerna som man vet kommer att gå igenom, man testar inte delar som skulle kunna krascha. Men det skulle nog vara väldigt bra med automatiserade tester på system utan gränssnitt som bara behandlar och skickar data.

Vad tror ni kan vara utmaningarna med att testa mobila applikationer?
Antalet plattformar som ska stödjas och olika enheter, skärmstorlekar samt olika OS versioner. Sedan blir det en utmaning vad man ska stödja om man utvecklar en mobilapplikation vilket leder till att kraven blir viktigare eftersom man där talar om vilka enheter och modeller som stöds. Än så länge verkar det inte vara lika mycket test kring appar, det kan bero på att det fortfarande anses som ”nytt” och är nog fortfarande lite omoget.

Designar ni apparna själva?
Det är lite olika, det beror lite på kunden. Vissa applikationer vill de ha helt färdiglevererade medan vissa har de någon webbyrå som designar.

Appar har ju funnits ganska länge men tror ni att det kommer bli mer företagsspecifika appar i framtiden?
Ja det är nog dit det är på väg, att använda det mer som ett verktyg i verksamheten. Appar som fungerar ihop med andra system i verksamheten.

5 Intervju med Systemutvecklare, Triona (Företag D)

Vilken typ av app är Tracs Flow?
Tracs Flow är en hybrid-app som är byggd på Cordova-plattformen.

Vilken funktionalitet finns i appen?
I appen kan man se pågående ordar, ändra orderstatus, signera en order, få push-notifieringar och använda sig av en GPS.

Vilka funktioner är beroende av enheten?
GPS samt Push-notifieringar. Åtminstone något som kan ge ett push-id tillbaka. När vi kör på webben så fejkar vi ett id som skickas in, så håller den reda på aktiva sessioner mellan enheter. Man ska kunna vara aktiv både på iPaden och telefonen och dessa ska synkas. (Till push-notifieringarna finns ett plugin och till GPS:en finns en plattform via webben som har en kartdekal.

Vilka är det som jobbar med appen idag?
Själva kodningen bakom appen så är vi idag två stycken som jobbar med.

Vilka modeller fungerar appen i?
I avtalet står de senare versionerna av iPhone och iPad med samt Nexus-plattorna och Nexus-telefonerna från fyra och framåt. Vi har dock inte testat på någon Nexus 6 ännu men vi antar att det fungerar eftersom jag har Android 5 även i min Nexus-telefon vilket fungerar bra. Officiellt supportar vi inte Samsung, men det är en stor användargrupp som har Samsung. Ibland får vi in ärenden att något inte fungerar och då försöker vi åtgärda det.

Hur har det blivit så, att ni inte supportar Samsung?
Den här (Nexus) kör stock Android och är orörd och iPhone är inte så mycket annat att välja på och skiljer sig inte så mycket. Samsung har en egen webb som den körs mot och använder sig av Firefox-derivator vilket kan ställa till det lite med animeringar i positioneringar, mer gränssnitt alltså. Det är sällan en funktion blir skillnad utan det är mer en upplevelse eller ett gränssnitt eller en knapp (eventhantering) som går fel, vilket också är det de kontaktar oss för i så fall.

Har ni fått några ärenden där pushen inte har fungerat för några modeller eller finns funktionen så att den fungerar för alla modeller?

Som jag minns så har pushen fungerat likadant alltid och vi har inte fått in några ärenden gällande det. Det var dock någon version i början som det var lite struligt med och vi fick ändra i native-koden och rätta det. Det gällde faktiskt iPhone eftersom IOS 6 och 8 initierar pushen på olika sätt så där fick vi faktiskt göra om det lite. Men nu med nya uppgraderingen av push-pluginet till Cordova så har vi fått samma beteende och har samma kodbas.

Det kanske är svårt att motivera t ex Olle, 62 år, som använder appen, att han måste köpa den nyaste iPhone för att få appen att fungera?
Ja, så är det. Vi måste ju rekommendera företaget att köpa en viss modell i början. Det är en

känslig fråga såklart då det handlar om pengar och bekvämlighet för kund så istället har det blivit att vi får ta den smällen att det ibland kan komma en Samsung-telefon där det skiljer sig när det gäller framförallt användarupplevelsen.

Har det varit några skillnader i modeller som har samma Android-version, t ex en Sony & och en Samsung?
Ja, det har det varit eftersom dom (Samsung) lägger in sitt eget vilket tyvärr gör att det blir skillnad.

På vilket sätt görs testerna idag?
Idag testar vi så att vi har ett Excel-ark där vi har testfall uppskrivet och någon enskild person går igenom de olika testerna. Vi är flera stycken som testar, beroende på vilka som har tid.

Vad är det som testas idag under testerna?
Man loggar in och testar resurser och hämtar hem ordar, ökar och minskar status på en order, signerar en order, ändrar egenskaper på en order, provar att installera och avinstallera appen på olika telefoner. De telefoner som man har till hands då alltså.

Lånar ni då olika mobiler från folk här på jobbet för att testa?
Ja, i alla fall på de mer kontrollerade testerna. Där är det noga att man testar på de enheter som man har i avtalet. Det kan jag dock säga att det kanske inte alltid täcks till hundra procent ändå för det är inte alla som har just de enheterna som finns i avtalet tyvärr.

Hur ofta genomför ni tester idag? Hur ofta uppdateras appen?
Det är egentligen i samband med när vi är på väg att släppa en version och det är ungefär varannan månad. Men låt säga att jag har ett ärende att vi ska in med manuell sortering i appen så får jag testa på så många enheter som jag kan under utvecklingens gång. Det är ganska ostrukturerat men det är så det går till. Framförallt så sitter jag på webben och gör det eftersom då får jag fram debug-utskrifter också och kan se om det blir något fel. JavaScript ger ju inga fel förrän man kommer över och exekverar, beroende på var felet ligger. (Ligger det på domen så kan det ge fel direkt) Oftast måste man dock gå in och göra det, vilket är det som blir det svåra.

På vilket sätt dokumenteras testerna?
Resultatet skrivs in i Excel, mestadels bara ja/nej på hur de olika funktionerna fungerar. I bästa fall någon kommentar om det.

Använder ni något speciellt testverktyg till testerna idag?
Nej, det gör vi inte. Utan det är att vi själva använder appen och försöker få fram felen på så vis.

Vilka är de största problemen med att testa just appen?
Det är just det att det är svårt att automatisera eftersom det kräver att man trycker på knapparna och navigerar runt. Det är svårt att testa installationsförfarandet.

Vilka delar tycker du är de viktigaste delarna att testa i appen? (Om man skulle automatisera detta)
Jag tycker att det skulle vara viktigast att testa själva synkroniseringsprocessen internt i appen. Man vill se att orderförändringarna fungerar bra. Det skulle vara bra om man kunde ”lura” den

och förse den med data och utifrån det se vilket beteende man får eftersom den ska bubbla upp automatiskt och t ex färga en rad eller ändra en siffra. Man vill kunna se om man skickar in något och får ut det som förväntas dvs. när man nått upp i alla Corbax hela vägen så ska vi ju kunna se högst upp om vi fick ut det vi förväntade oss, t ex DE på orderstatus.

Status och orderinnehållet är ju det viktigaste samt att den kommer ut i bra tid. Sen om pushen inte alltid fungerar kan även bero på tjänstesidan och inte från appens sida. Det viktigaste är att vi kan starta och registrera och få ett push-id från enheten. Om vi inte skulle få det så kanske vi kan ha något strul med certifikaten, signeringen eller att något med bygget gått fel alternativt uppstart av Angular eller Cordova.

Det hände någon gång i början att vi hade fel med timing i JavaScripten där den inte kunde serva oss med ett app-id när vi behövde ha den så det är också sådana saker som man kan åka på och som kan vara svåra att testa.

Har ni kollar något på de olika alternativen för testverktyg som finns idag?
Jag kan inget namn på de olika verktygen men jag har för mig att vi kanske gjort en liten sökning om det. I demonstrationssyfte så har vi använt oss av Genymotion-emulatorn.

Finns det någon del som är extra tidskrävande som du tycker att vi ska fokusera på och kolla på när det gäller automatisering?
Jag tycker nog att ni kan fokusera på att föda appen med exempeldata, nästan som ett band för appen att nu kommer det in det här och det här och den här förändringen gjordes på ordern för det ger utslag på alla olika beteenden som den har och som ska hanteras. Så det är något som man skulle kunna göra och ha som en approach. De ramverk som jag har hört talas om, ska väl kunna ge ett klick på en viss knapp dvs. trigga en viss klickhändelse. Skulle ni kunna automatisera och välja att fylla i t ex två rutor och klicka på att logga in så vore det väldigt bra.

6 Mailintervju Testare, Företag E

Berätta kort om ert företag!
Vi är ett konsultbolag med runt 100 anställda på 4 orter i Sverige. Stockholmsbolaget är störst. Inom koncernen ingår även ett utbildningsbolag som främst utbildar inom Test och Krav.

Har ni utvecklat mycket mobila applikationer? Vad har ni huvudfokus på i ert företag?
Vi har ingen egen utveckling av mobila applikationer men vi har en hel del konsulter som sitter i uppdrag där det byggs mobila applikationer. Test och krav som fokusområden.

Utvecklar ni till en specifik plattform?
I vårt nuvarande uppdrag så utvecklas appar för iOS, Android och Windows.

Vilken typ av applikationer utvecklar ni? (Native, Hybrid eller mobila webbapplikationer)
Native appar.

Har ni något speciellt testverktyg som ni använder?
För aktuellt uppdrag används en kombination av Test Manager och Team Foundation Server.
Om inte, varför inte och vet ni något om de olika alternativen för testverktyg som finns idag?

På marknaden finns en uppsjö av testverktyg att tillgå. Beroende på kund och uppdrag så kan man säga att de ”större” kunderna oftast arbetar i ett väletablerat testverktyg såsom QC eller TFS. Uppföljning och rapportering är något som är en viktig del i ett fullvärdigt testverktyg.

Om ja, till vilken typ av tester använder ni testverktyget?
Till de acceptanstester som vi utför.

Gör ni testerna manuellt eller automatiserat? Om ni testar allting manuellt, på vilket sätt tror ni att automatiserad testning skulle kunna vara till en fördel?
Då vi som svarar på denna fråga sitter i acceptanstestfasen utförs alla tester manuellt. Att använda automatiserade tester vid acceptanstester är inte något vi rekommenderar. Under systemtest är det däremot något som man bör använda sig av. Problematiken kring att täcka ett oändligt antal kombinationer av enheter och operativsystem är nog den del där vi ser störst nytta med automatiserade tester. En annan fördel är att kunna täcka regressionstester kring huvudflöden som behöver testas ofta.

Vilka är de största utmaningarna med testning av mobila applikationer?
Den största utmaningen är utan tvekan den ständigt växande skaran av enheter och operativsystem. Konsten att välja ut en bra kombination av enheter är nyckeln till framgång. Lägg sen till uppkopplingar, skärmstorlekar och kameror så ser man snart nyttan av smarta urval.

Vilken av delarna tror ni är i störst behov av att automatiseras? Vilken del tror ni är svårast att automatisera?
Som tidigare beskrivet i punk 6 är möjligheten att snabbt kunna täcka in en stor del av telefoner och OS versioner något som skulle underlätta. Vi tror också att just detta är det svåraste att lösa med automatisering.

Regressionstester kring flöden som ofta är påverkade är något som bör automatiseras om det är möjligt. Det lättaste att automatisera kan vara de anrop som apparna kör mot appservern.

Man kan aldrig automatisera en ”känsla” eller en ”upplevelse”.

På vilket sätt dokumenteras testningen?
Testplan, testfall, statusrapporter, felrapporter, testrapport och releaseuppföljning.

7 Mailintervju Utvecklare, Företag F

Berätta kort om ert företag! Har ni utvecklat mycket mobila applikationer? Vad har ni huvudfokus på i ert företag?
Företaget är ett mindre IT-företag och vi utvecklar appar till mobiler och webben. Dels några egna projekt men framförallt på uppdrag och som konsulter. Företaget startades 2011 och vi är nu 5 anställda.

Utvecklar ni till en specifik plattform?
Vi började med enbart iOS men har utökat till Android och har även experimenterat med Windows Phone 8. På senare tid har vi jobbat mer med webb-applikationer.

Vilken typ av applikationer utvecklar ni? (Native, Hybrid eller mobila webbapplikationer)
Både native och mobila webbapplikationer beroende på vilka krav som ställs på produkten samt kundens önskemål.

Vi hörde att ni använder TestFlight som testverktyg. Hur föll det sig att ni började använda just det verktyget? Har ni testat något annat verktyg också?
När vi gjorde en app för ett stort appföretag, använde de Testflight till att snabbt distribuera appen till sitt test-team. Vi tyckte det var väldigt smidigt och använde det senare på en egen app. Sedan dess har dock testflight blivit uppköpt av Apple och kan inte längre användas för android-appar så på den senaste appen vi utvecklade för Android fick vi klara oss utan. Vi har börjat titta på alternativ men inte fastnat för något ännu. Crowdtesting är på frammarsch och skulle kunna vara ett alternativ för framtiden.

Till vilken typ av tester använder ni TestFlight? Några speciella fördelar/nackdelar som ni skulle tycka är värda att nämnas gällande TestFlight?
Vi använde TestFlight till att enkelt och snabbt distribuera nya builds till flera devices så det var framförallt manuella tester det rörde sig om. Vi gillade att det var enkelt och smidigt att sätta upp samt att det fanns stöd för continuous deployment, dvs att det gick att ställa in så att en ny version lades upp så fort man checkade in ny kod.

Gör ni testerna manuellt eller automatiserat? Om ni testar allting manuellt, på vilket sätt tror ni att automatiserad testning skulle kunna vara till en fördel?
Vi är övertygade om att man behöver både och. Automatiserade tester är grunden man står på som ger en trygghet i att man inte bryter gammal funktionalitet när man implementerar nytt. Manuell testning behövs för att det alltid finns saker man inte tänkt på och framförallt GUI:t kan vara svårt att skriva automatiska tester för.

Vilka är de största utmaningarna med testning av mobila applikationer?
Helt klart fragmenteringen av devices för Android (iOS är på väg dit också) om man kör native och gamla webbläsare som inte uppfyller Html5-standarden om man kör webb-baserade appar. Men inte heller alla nya webbläsare stöder allt så där blir det lätt lite problem. Att göra ett GUI som passar på alla olika skärmstorlekar och upplösningar är en stor utmaning och är svårt/tidskrävande att testa. Vi kan omöjligen ha en av varje android-device som tillverkats så vi kan aldrig vara säkra där att det fungerar perfekt överallt.

Vilken av delarna är i störst behov av att automatiseras enligt er? Vilken del tror ni är svårast att automatisera?
All affärslogik måste testas via enhetstester. Integrationstester är också viktiga då vi ofta har

en/några som jobbar på servern och andra på klienten så de blir till en sorts kontrakt som både sidor måste följa. Vi använder de bl.a. till att specificera ett REST-API. Svårast att testa automatiskt är nog GUI:t. Det är möjligt att göra men vi känner att det sällan är värt det då det är väldigt många saker som behöver testas och design och layout oftare ändras under utvecklingstiden.

På vilket sätt gör ni enhetstesterna och integrationstesterna?
På lite större projekt använder vi byggskript som kör alla tester så fort vi checkar in kod och tillåter inte att kod som inte går igenom hamnar på servern. Vi använder Jenkins för detta. På mindre projekt händer det att kunden inte vill lägga tid på tester utan snabbt vill ha en prototyp som vi sen arbetar vidare på. Kvalitén på koden blir då inte lika bra men för icke affärskritisk verksamhet kan det fungera.

På vilket sätt dokumenteras testningen?
Vi dokumenterar väldigt lite kod och det inkluderar test-kod. Om kunden efterfrågar det ger vi siffror på testtäckning men de siffrorna känner vi själva inte är särskilt relevanta. Vi ser testningen som en del av utvecklingen och den är till för oss själva framförallt och inget som kunden borde behöva bry sig om. Den stora vinning är snarare i att det får oss att skriva bättre kod, snarare än att hitta buggar.

8 Testfall

System: Tracs Flow

Testfall	Teststeg	Förväntat resultat	Storyboard nr.
T1	1. Starta appen 2. Fyll i användarnamn och lösenord 3. Tryck på Logga in	1. Appen startas 2. Uppgifterna fylls i korrekt 3. Om uppgifterna stämmer så ska man loggas in.	Inloggningssida
T2	1. Välja resurs 2. Trycka på klar	1. Resurserna ska visas 2. Välkomstsidan ska visas	Resurssida
T3	1. Trycka på knappen orderlista 2. Trycka på en order 3. Trycka på bakåt-knappen	1. Alla ordrar ska visas 2. Detaljerad information om ordern ska visas 3. Alla ordrar ska visas	Startsida, ordersida
T4	1. Trycka på knappen orderlista 2. Ändra orderstatus	1. Orderlistan ska visas 2. Orderstatusen ska ändras	Startsida, ordersida
T5	1. Gå till manuell sortering i menyn 2. Ändra ordning på ordrar	1. Manuell sortering ska visas 2. Ordrar visas i ny ordning	Ordersida
T6	1. Tryck på Lediglista 2. Ändra status till kör	1. De olika statusalternativen visas 2. Statusen ändras till kör	Startsida, lediglista
T7	1. Tryck på knappen orderlista 2. Tryck på en order 3. Visa till-adressen i Google Maps	1. Alla ordrar ska visas 2. Detaljerad information om ordern ska visas 3. Adressen visas i Google Maps	Startsida, ordersida, Google Maps
T8	1. Logga ut	1. Användaren loggas ut och inloggningssidan visas	Startsida, inloggningssida

System: Testverktyg - Funktionella krav

Testfall	Testfråga
TV1	1. Spela in testfallen 2. Köra inspelat testskript på flera enheter
TV2	1. Testa ett testskript på flera enheter samtidigt

System: Testverktyg - Icke funktionella krav

Testfall	Frågor på Icke funktionella krav
TV1 IF	Var installationen av testverktyget enkelt?
TV2 IF	Är testverktyget användarvänligt?
TV3 IF	Finns det dokumentation och bra användarmanualer?
TV4 IF	Är testverktyget möjligt att installera på flera datorer och användas av flera användare?