Karlstad Business School
Handelshögskolan vid Karlstads universitet

Marcus Sjölin

Att automatisera funktionstester fokuserat på e-handelssystem

En fallstudie på Askås I&R med fokus på testprotokoll, programspråk och effektiviseringsgrad

To automate functional tests focused on e-commerce systems

A case study at Askås I&R focusing on test protocols, programming languages and efficiency levels

Informatik
C-uppsats

Termin: VT-17
Handledare: John Sören Pettersson

Karlstad Business School
Karlstad University SE-651 88 Karlstad Sweden
Phone:+46 54 700 10 00 Fax: +46 54 700 14 97
E-mail: handels@kau.se www.hhk.kau.se

Abstract

För att minska mängden fel och öka pålitligheten i ett IT-system är testning något som bör utföras. IT-företaget Askås I&R använder sig i dagsläget av stora mängder manuella tester. För att effektivisera testprocessen ska automatiserade tester införas. Fokus i denna studie ligger på att undersöka om programspråk bör väljas för att passa testverktyg eller om testerna bör anpassas efter programspråket som används frekvent på företaget, samt hur mycket arbetstid företag kan tjäna på att automatisera sina testfall.

En översiktlig undersökning gjordes för att undersöka vilket verktyg som är bäst lämpat för Askås I&R baserat. För att utvärdera vilket programspråk som är relevant för företaget användes en enkätundersökning bland företagets testare samt en öppen intervju med en systemarkitekt på företaget.

Fem av företagets arton testprotokoll automatiserades. Tidsåtgången för de manuella och de automatiserade testprotokollen jämfördes och analyserades. Utfallet tyder på att det finns mycket tid att spara på automatisering. I uppsatsens slutsatser framhålls att automatisering av testfall bör prioriteras efter vissa kriterier vid utvecklingen av nya systemversioner.

Nyckelord: automatiska funktionstester, Selenium, testfallsprioritering, effektiviseringsgrad, node.js

Förord

Jag vill tillägna ett stort tack till min handledare John Sören Pettersson som bidragit med hjälpande information och konstruktiv kritik under skrivandet av denna uppsats. Jag vill dessutom tacka Universitetslektor Martin Blom för han bidrag i form av samtal, samt utlåning av passande litteratur till studien. Jag vill också tacka de anställda på aktiebolaget Askås I&R för deras medverkande för datainsamling till studien, intresse i uppsatsen, samt insikt i utvecklingen av de automatiska testfallen som varit fokus under uppsatsen.

Innehållsförteckning

1. Inledning

I det inledande kapitlet så beskriver jag först problemområdet angående dagens komplexa IT-system, dess roll i dagens samhälle, samt hur man får system att bli pålitliga. Här ger jag också en kort introduktion av fallstudiens företag i fråga. Efter problemområdet så följer en beskrivning av studiens syfte, sedan en avgränsning av studien, följt av en presentation av målgruppen i fokus och kapitlet avslutas sedan med relevanta undersökningsfrågor som kommer att ligga i fokus under studien.

1.1 Problemområde

Dagens moderna samhälle använder sig av allt fler system som hjälper oss att bedriva vardagen. Dessa system är allt ifrån mobila applikationer och webbsidor, till större system som driver industrier och större företag samt myndigheter. Det är inte så svårt att föreställa sig varför man i hög grad måste lita på dessa system och varför de måste fungera näst intill felfritt.

Men hur gör man då för att få ett system som man kan lita på? Det finns trots allt inget perfekt system som alltid kommer att fungera felfritt, dygnet runt, året om. Det är här som testning kan finslipa system till nästintill felfritt. Något man bör ha i åtanke är att testning, likt systemet i sig, är i en form av kontinuerlig utveckling. Då ett system underhålls och uppdateras så ändrar det ständigt förutsättningarna för alla de testfall som har skapats till systemet. Detta betyder i sin tur att alla testfall också kräver ständigt underhåll för att kunna ge relevanta resultat när de används – testning och testfallsutveckling är kontinuerliga aktiviteter.

Webbsidor har blivit en stor del av vardagen och allt fler personer använder sig av internet för att utföra sina ärenden, då en webbtjänst som Facebook hade drygt 1.86 miljarder aktiva användare i månaden, från och med december 2016 (Facebook 2017). Dessa system måste vara byggda av hög kvalitet för att kunna stödja den enorma mängden användare, samtidigt som exempelvis säkerheten av personuppgifter behålls intakt. Det är här som testning blir högst användbart! Programvarutestning har alltid varit en vital del av systemutveckling och har likt allt annat utvecklats med tiden. Idag testas komplexa system med hjälp av automatiserade tester för att effektivt och snabbt kunna hitta fel i systemet, som därefter kan rättas till. Att automatisera testfall innebär dock en tidskonsumtion som innefattar uppbyggnad av dessa testfall. På grund av detta måste testaren prioritera vilka testfall som innebär störst vinst i arbetstid.

Aktiebolaget Askås I&R Internet & Reklambyrå är ett företag som bedriver samt underhåller ett så kallat CMS (Content Management System) till andra företag som vill bedriva webb-baserad e-handel. Askås I&R levererar helt enkelt en komplett e-handelsplattform som är fullt utrustat med betalning, logistik, design, marknadsstrategi, administration och mycket annat. (Askås I&R 2017). Askås I&R har vuxit med rasande takt på senare år men använder sig fortfarande av stora mängder manuella tester för att testa systemtillägg på sitt system och är nu ute efter att effektivisera sin testning med hjälp av automatiserad testning.

En komplicerande faktor är att företaget baserar sitt system på ett programspråk som saknar stöd på majoriteten av dagens verktyg som används för automatiserad testning. På grund av detta så är det av intresse för företaget att undersöka vilket alternativt programspråk som passar bäst för just Askås I&R. Automatisering av testfall innefattar alltså en uppbyggnads-process som är tidskrävande inte bara för att det handlar om vilka testfall samt testprotokoll som man bör prioritera att automatisera utan också om val av verktyg och programspråk måste göras utan också för att problemområdet även

1.2 Syfte

Syftet med uppsatsen, samt denna fallstudie, är att undersöka samt implementera ett passande system för automatiserade funktionstestning till aktiebolaget Askås I&R.

1.3 Avgränsning

Då det finns mängder med verktyg som möjligtvis uppfyller den roll som Askås I&R eftertraktar på marknaden, så måste studien avgränsa sig till att använda ett specifikt verktyg och sedan optimera verktyget till Askås I&Rs testrutiner. Tillsammans med en kurskamrat, som samtidigt bedriver en studie av automatisering av tester hos Askås I&R, har jag därför endast gjort en översiktlig undersökning av några av de verktygen som finns och sedan motiverat till valet. Detta betyder att undersökningens fokus inte kommer att ligga i vilket verktyg som bäst passar sig för företaget, utan att välja ett pålitligt verktyg och sedan tillämpa och optimera det till Askås I&R för att kunna mäta effekter.

1.4 Målgrupp

Målgruppen för detta examinationsarbete är främst Aktiebolaget Askås I&R. Andra företag och organisationer som liknar aktiebolaget Askås I&R, som är intresserade av att implementera automatiserade funktionstester på sitt system, kan också finna denna studie intressant. Fokus i studien ligger hos automatiserade funktionella tester, det vill säga tester som simulerar en verklig användare. Detta betyder att man kan finna arbetet intressant om man har ett intresse inom funktionstester, eller om man vill skapa en simulering av en person på internet. Självklart så kan också arbetet vara av intresse till andra studenter, företagare, myndigheter och webbutvecklare som vill få en grundlig genomgång av hur man bedriver tester i dagens moderna samhälle.

1.5 Undersökningsfrågor med tillhörande förklaring och motivering

u1.

Vilka testprotokoll är relevanta att automatisera för företaget?

Att automatisera ett testfall inom ett testprotokoll kan resultera i stora besparingar av resurser, i både form av tid och pengar. Precis hur mycket man sparar på en automation beror uppenbarligen på vilket testfall som blir automatiserat. Vissa testprotokoll, med tillhörande testfall, kommer realistiskt att utföras vid var uppdatering av systemet, medan andra testprotokoll endast kommer att användas vid specifika uppdateringar. Studien bör alltså prioritera samt undersöka vilka testfall samt testprotokoll som resulterar i störst vinst för företaget.

u2.

Vilket programspråk för automatiska funktionstester är mest lämpligt för Askås I&R?

För närvarande bygger Askås I&R sitt system på programspråket Perl. En nackdel med detta är att Perl saknar popularitet hos utvecklare idag, vilket också innebär att verktyg för automatiserade funktionstester inte är avpassade efter Pearl. Denna undersökningsfråga är relevant då det kan vara intressant för företaget att utveckla automatiserade funktionstester i ett verktyg som använder sig av ett programspråk som åtnjuter större stöd från verktyget. Ett delsyfte av denna uppsats blir alltså att undersöka vilket verktyg, samt vilket programspråk som är bäst lämpat för företaget.

u3.

Till vilken grad effektiviseras testrutinerna av automatisering av testfall?

Är det relevant för företaget att implementera och underhålla dessa automatiska funktionstester ur ett tidsbesparande perspektiv? Fokus för denna undersökningsfråga ställs i denna studie ur ett tidssparande perspektiv.

1.6 Centrala begrepp

Inom rapporten så används ett antal centrala begrepp för att kommunicera metoder och tillvägagångssätt. Då dessa oftast har en nyckelroll till att förstå sammanhanget så är det viktigt att läsaren har en bra uppfattning av vad dessa begrepp innebär. Centrala begrepp beskrivs nedan.

1.6.1 Content Management System (CMS)

Ett Content Management System, även kallat CMS, är ett så kallat innehållshanteringssystem som används för att redigera och underhålla information på en webbsida. Det finns olika typer av CMS som företag, privatpersoner samt organisationer kan välja mellan, beroende på vilken budget samt kunskap man besitter och till vilket användningsområde man utgår ifrån.

Webbplatsen *Interactive Design* (ITD 2017) beskriver i en artikel några av de mest populära CMS som används idag. Artikeln delar in systemen i två kategorier, de kommersiella och de som är öppna (open-source).
De kommersiella CMS är de som säljs kommersiellt och då används genom en licens. Dessa system har ofta en större kapacitet till att hantera mer data och besitter generellt sett mer funktionalitet än de öppna systemen. Några av de mest populära kommersiella CMS som används idag innefattar bland annat EPiServer, Escenic, PoloPoly och Roxen. (ITD 2017).

De öppna CMS är de som finns tillgängliga för alla. De har alltså öppen funktionalitet och källkod som gör att alla har full tillgång till att utnyttja systemet. Detta betyder att varje individ har tillgång till att modifiera systemet och dess utseende till individens önskemål. Öppna system besitter dock allmänt mindre funktionalitet jämfört med de kommersiella systemen, då de generellt sett utvecklas i ett långsammare tempo på grund av att systemet ej besitter någon budget. Några av de mest populära öppna CMS som används idag innefattar bland annat WordPress, Joomla!, Mambo och Drupal. (ITD 2017)

Det CMS som Askås I&R utvecklar är därav ett kommersiellt CMS som specialiserar sig på nätbutiker och annan övrig e-handel. Detta betyder att deras system är speciellt anpassat för att kunna hantera relevant data, som artiklar och kundregister, på ett simpelt sätt så att kunden enkelt och effektivt kan utnyttja systemet till dess fulla potential.

1.6.2 Testprotokoll

Ett testprotokoll är ett uttryck som Askås I&R använder sig av för att beskriva ett ärende i behov av testning. Ett sådant testprotokoll består främst av en specifikation, vilket beskriver översiktligt vad som skall testas, samt minst ett testfall (se 1.6.3 *Testfall*), vilket beskriver punktligt sekvensen som skall testas. Ett testprotokoll är då en sammanställning av ett eller flera testfall som utförs för testning av system. Ett testprotokoll i studiens omfattning innebär de valda testfallen som undersökningen fokuserar på.

1.6.3 Testfall

Ett testfall beskriver mer sekventiellt vad som bör utföras för att testprotokollet (se 1.6.2 *Testprotokoll*) skall resultera i något bärande. Ett testfall illustreras ofta i någon form av lista som kan ‘bockas av’, för att stegvis visa när testprotokollet är färdigställt. Ett testfall är en uppsättning instruktioner med tillhörande förväntat resultat som en testare utgår ifrån. Avvikelser i det förväntade resultatet är av intresse då detta tyder på oväntat utfall från systemet. För definition av testfall, se 2.1.3 *Testfall*.

1.6.4 Regressionstestning

Begreppet regressionstestning som beskrivs under; 2.3 Testfallsprioritering, är testning av hela, eller delar av ett system vid tillämpning av ny funktionalitet, vilket Askås I&R praktiserar vid ny systemversion.

2. Att utföra tester

Detta kapitel beskriver vad testning innebär, dess syfte och är menat att ge en grundlig förståelse om vad ett testprotokoll samt ett testfall innebär. Jag beskriver också vad man bör prioritera att automatisera, då att automatisera hela testningssektionen på en och samma gång är orimligt på grund av hög tidskonsumtion. Utöver detta så diskuterar jag kring varför, när och till vilken grad man bör automatisera sina testfall.

I denna undersökning utgår jag från tidigare undersökningar, artiklar samt konferensrapporter, där jag hittat liknande studier inom systemtestning.
Litteraturval till undersökningen är främst tidigare undersökningar som bedrivits för just automatiserade funktionstester på webbplatser. Då detta är ett mycket specifikt område så användes också litteratur som fokuserar på funktionstester, samt systemtester. Sådan litteratur är intressant då en webbplats också i grunden är ett system, som bygger på liknande struktur och möter liknande hinder som ett traditionellt IT-system; se 2.1 Öppen *intervju med Universitetslektor Martin Blom på Karlstads Universitet*. Standardlitteratur angående system- och funktionstester har föreslagits av Martin Blom. Det utkommer hela tiden nya böcker i ämnet (efter att detta kapitel skrivits kom Tarlinder 2017), men de innehåller inte mycket nytt utan är snarare nya sätt att presentera redan etablerade idéer. Även annan litteratur angående tester, som funktionstester, är av intresse då de ger en bättre bild av varför automation kan vara ett bra alternativ för företaget i fråga. Men det här kapitlet refererar endast till rekommenderad litteratur av Blom.

Att hitta relevanta undersökningar som utvärderar verktyg som utför automatiserade funktionstester på webbaserade system, har visat sig vara något av en utmaning. På grund av detta utgår stora delar av undersökningen av potentiella verktyg från facklitteratur på internet. Facklitteratur som har varit av intresse är framförallt webbsidor som utvärderar verktyg för automatiserade funktionstester på webbaserade system, en jobbannons som utför tester på system, andra företags val av verktyg, samt en konferensartikel.

2.1 Öppen intervju med Universitetslektor Martin Blom på Karlstads Universitet

Den öppna intervjun inleder teorikapitlet med anledning av att den ger en mycket bra grund för läsaren. Blom diskuterar bland annat hur webbshoppar kan kopplas till vanliga IT-system, hur automatiska testfall bör hållas dynamiska, problemområdet med att använda ett relativt ovanligt programspråk som Perl och hur en testares roll påverkas utav automation. Intervjun ägde rum den 10 mars 2017.

Syftet med intervjun var att få en kvalitativ åsikt och synvinkel angående automatisering av webbaserade funktionstester, och utifrån detta välja relevant litteratur att utgå från. Intervjun inleds med en diskussion om hur webbaserad testning förhåller sig till vardaglig testning av system. Blom förklarar att webbshoppar är ett system likt andra system, och att testning av funktionalitet på sådana system ligger på en mer generell nivå. Som testare bör man ha i åtanke att syftet med funktionell testning är att testa hur systemets delar interagerar med varandra, och på så sätt testa systemet i helhet. Samtalet fortsätter sedan med att Blom

förklarar att GUI (Grafisk Användargränssnitt) oftast kräver manuell testning, då webbsidors innehåll ofta byter plats, baserat på vilken skärmstorlek, upplösning på bildskärmen eller zoom-styrka som användaren utnyttjar. På grund av detta bör testaren bygga automatiserade testfall på ett dynamiskt sätt, genom att effektivt hitta specifika element på ett dynamiskt sätt. Ett exempel på detta är att “klicka” på en knapp genom att utnyttja dess element-id, och inte “klicka” på ett specifikt ställe på skärmen.

Intervjun fortsätter sedan med att diskutera hur programspråket Perl förhåller sig till den absoluta majoriteten av verktyg som används för automatisering. (se.2.4 Verktyg för automatisering). Blom förklarar att olika programspråk oftast är bra på olika saker, men att om officiellt stöd saknas för programspråket, så kan det vara av stort intresse för företaget att byta språk. Dock inte nödvändigtvis byta språk i helhet, men att i alla fall utöka kunskaper och lära sig ett alternativt språk för testning. Då testning inte skall integreras med det verkliga systemet, utan jobba isolerat från källkoden, så innebär det att programspråket för testning kan ändras utan framtida problem för det verkliga systemet. Blom förklarar också att då verktyg som Selenium officiellt utvecklar stöd för specifika programspråk, så innebär det även att de språk sannolikt utvecklas mer i framtiden. Utveckling av andrahands språk med en mindre användarkrets har oftast en långsammare utvecklingsprocess.

Samtalet fokuserar sedan på att diskutera angående hur testarens roll påverkas av automatisering, då syftet med automatisering av testfall är att eliminera utförande av manuella testfall. Blom reflekterar sedan kring att testning alltid bör uppdateras i samma grad som systemet. Testning är en kontinuerlig cykel som bör underhållas för att erhålla en effektiv testningsrutin. Ytterligare så innebär automatiserade testfall att testaren kan lägga sin värdefulla tid på mer relevanta uppgifter, såsom att effektivisera programkoden, underhålla programkoden, skapa fler automatiserade testprotokoll, samt lära sig mer effektiva metoder för testning av system. Målet med automatiserad testning är inte att besitta en annan form av testning, utan att slippa testa manuellt. Viss funktionalitet kan inte heller automatiseras, då viss funktionalitet kan kräva varierad input. Generellt så ska man försöka att automatisera det som upprepas.

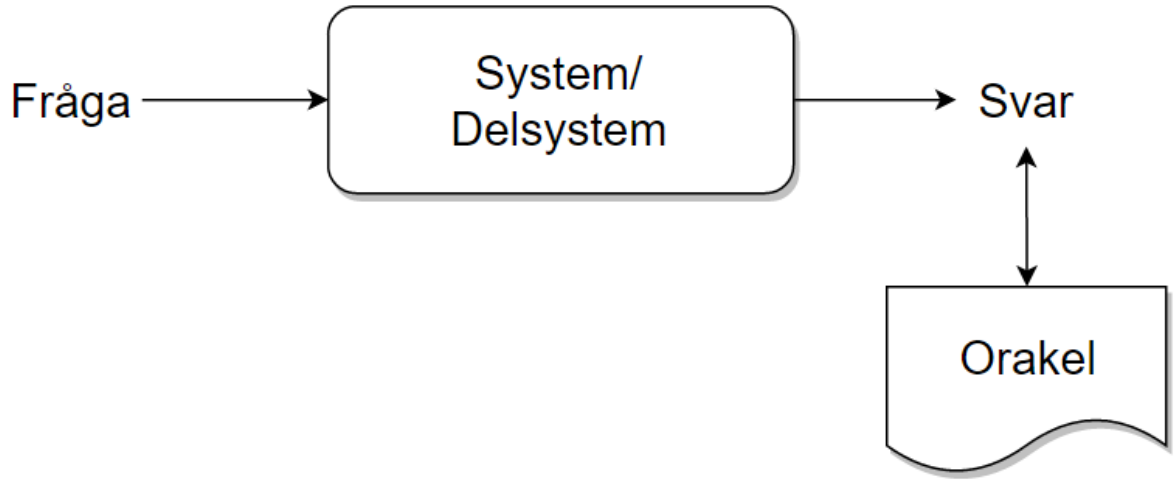
2.2 Att testa system

Att testa är viktigt. Detta är något som alla utvecklare kan intyga, vare sig utvecklaren jobbar med att bygga webbapplikationer, lokala system, högtalare, eller broar. Man säljer inte en produkt innan man har kvalitetssäkrat produkten, sedan hur väl man bestämmer att testa denna produkt är upp till utvecklaren. Detta kan man relatera till Askås I&R, de säljer en produkt i form av ett system som växer kraftigt i hand med att företaget expanderar med fler kunder. Större system betyder att allt fler defekter kan tillkomma i systemet, då fler delar måste integreras med andra.
Indirekt så kan man tänka att fler och större tester resulterar i en högre kvalitet av systemet och då högre kvalitet betyder färre defekter, så kan man förstå att kvaliteten betyder mycket för ett sådant företag.

García och Dueñas (2014:2) beskriver systemtestning som huvudaktiviteten för att utvärdera produktens kvalitet, och för att förbättra den, genom att identifiera defekter i system. Detsamma gäller också för testning av webbsidor och webbapplikationer, fast till en högre grad. Då webbsidor och webbapplikationer ofta utsätts för högre konkurrens än vad lokala

system gör. Detta på grund av att användare alltid förväntar sig att sådana system skall fungera felfritt varje gång det används, och om de ej gör det så går användaren vidare till en konkurrents system (Ammann & Offutt 2008).

Ryber (2007:33) beskriver testning som en process av att använda ett system eller ett delsystem under givna förhållanden, genom observation eller notering, och sedan utföra en utvärdering av detta system eller delsystem från ett givet perspektiv (se figur 1).



Figur 1. Författarens illustration av Ryber (2007:33) modell för hur man utför ett test.

Utifrån modellen för test (se figur 1) så förklarar Ryber att testning handlar om att ställa frågor och sedan utvärdera de svar man får mot något form av *orakel*. Oraklet är då det förväntade resultatet från testet. Ryber beskriver också att tester består av fyra centrala problem: hur vi ställer frågor, vilka frågor man bör ställa, hur man ser på resultatet, samt vad oraklet består av. De centrala problemen beskriver alla de motstånd man möter som en testare, då alla dessa centrala problem kräver..

2.2.1 Funktionstester

Myers (2012:119) beskriver ett funktionstest som en process för att finna avvikelser mellan ett program och de externa specifikationerna, då en extern specifikation är en precis beskrivning av ett programs beteende från en användares perspektiv.
Myers beskriver också att ett hypotetiskt system som är färdigställt består av upp till 5% felaktig kod (2012:137-8) och att man realistiskt sett kommer att hitta 98% av alla fel som finns i systemet. Av dessa fel så är ungefär 30% sådana som relateras, och förhindras, med hjälp av funktionella tester. Ett funktionstest är i grunden en black-box aktivitet, när det ej utförs på mindre program (2012:119). Ammann & Offutt (2008:21) definierar black-box tester som härledande tester från externa beskrivningar av mjukvaran, vilket inkluderar en specifikation, krav för utförande, och design.

2.2.2 Testfall

Ammann & Offutt (2008:15) definierar ett testfall som att vara uppbyggt på testfalls-värden, förväntade resultat, före-värden (prefix), och efter-värden (postfix), som alla är nödvändiga

för att utförande och utvärdering av mjukvara under testningsprocessen. Ryber (2007:27) beskriver att en testares mål är att utföra ett antal testfall, och utifrån resultaten från dessa tester, besvara om krav på systemet nås eller ej.

2.3 Testfallsprioritering

Universitetslektor Blom (se 2.1 *Öppen intervju med Universitetslektor Martin Blom på Karlstads Universitet*) förklarar att de testfall som oftast utförs bör prioriteras att automatiseras. Detta på grund av att det är de testfall som oftast utförs av testaren, och på så vis bidrar till störst tidskonsumtion för testning. Rothermel et al. (1999) beskriver syftet med prioritering av testfall i samband med regressionstestning (se 1.6.4 *Regressionstestning*), som att utföra testfall i den ordning som resulterar i högst sannolikhet att hitta ett fel. Att prioritera testfall är effektivt vid ett flertal scenarier, vilket inkluderar:

1. Om testaren önskar att lokalisera ett flertal systemfel tidigt i utvecklingen.
2. Om testaren prioriterar att lokalisera kritiska fel tidigare i testningsprocessen.
3. Om testaren önskar att hitta fel relaterade till testning av koden.
4. Om testaren önskar att testningsprocessen skall utföras under ett högre tempo.
5. Om testaren önskar att öka pålitligheten hos systemet under ett snabbare tempo.

Rothermel et al. (1999:2-3) beskriver nio olika prioriteringstekniker för testfall. Dessa tekniker inkluderar: *No prioritization*, *Random prioritization*, *Optimal prioritization*, *Total branch coverage prioritization*, *Additional branch coverage prioritization*, *Total statement coverage prioritization*, *Additional statement coverage prioritization*, *Total fault-exposing-potential (FEP) prioritization*, och *Additional fault-exposing-potential (FEP) prioritization*.

No prioritization utgår från att utföra testfall utan någon specifik prioritering. Utfallet av en sådan teknik, kan variera beroende på hur tekniken var konstruerad.
Random prioritization utgår från slumpvalt prioriterade testfall.
Optimal prioritization utgår från vilka testfall som med störst sannolikhet förväntas resultera i att hitta fel, baserat på tidigare kända fel.
Total branch coverage prioritization handlar om att prioritera testfall baserat på hur stor del av systemet som testfallet täcker (branch). Testfall som täcker större delar av systemet prioriteras först.
Additional branch coverage prioritization utgår ännu en gång från hur mycket av ett system ett testfall täcker (branch). Dock prioriteras testfall också baserat på vilka delar av systemet de täcker, så att så stor del av systemet täcks som möjligt. Testfall prioriteras alltså efter storlek, och vilka delar av systemet de täcker.

Total statement coverage prioritization, prioriterar likt *Total branch coverage prioritization*, från hur mycket av ett system ett testfall täcker, dock denna gång i form av ett uttryck (statements).
Additional statement coverage prioritization, prioriterar likt *Additional branch coverage prioritization*, från hur mycket av ett system ett testfall täcker, dock denna gång i form av uttryck (statements).
Total fault-exposing-potential (FEP) prioritization skiljer sig från ett uttryck/statement och område/branch-tekniker, då dessa tekniker endast testar om dessa uttryck eller områden har

undergått testning. Denna teknik utgår från att testa om ett system misslyckas, om ett specifikt fel utlöses av ett specifikt fel hos ett uttryck.
Additional fault-exposing-potential (FEP) prioritization, likt de andra tekniker av typen *Additional*, utgår från att prioritera testfall så att så stor del av systemet täcks som möjligt. Likt här så utgår tekniken från att testa om ett system misslyckas, om ett specifikt fel utlöses av ett specifikt fel hos ett uttryck.

2.4 Verktyg för automatisering

2.4.1 Kriterier för verktyg

När det gäller att undersöka vilket verktyg som är mest relevant för just denna fallstudie så finns det ett antal kriterier som ställs av Askås I&R som verktyget bör uppfylla.

1. Först och främst så bör verktyget vara gratis av finansiella skäl. Av denna anledning så kommer verktyg av sorten open-source att prioriteras, då dessa verktyg är öppet utvecklade och därav gratis.
2. Ett andra kriterium som verktyget bör uppfylla är att det borde vara flexibelt när det handlar om vilket programspråk som kan användas för utveckling. Detta kriterium är grundat i dels att Askås I&R använder sig primärt av programspråket Perl vid utveckling, så erfarenhet finns där, och dels av att denna fallstudies syfte är att hitta ett alternativt programspråk som kan passa företaget, förutsatt att det programspråk stödjer verktyget mer än Perl.
3. Ett annat ytterst viktigt kriterium är att verktyget bör stödja majoriteten av moderna webbläsare. Detta höjer generellt kvaliteten på tester, då webbläsare hanterar information och annan relevant data på olika sätt. Genom att utföra identiska tester på olika webbläsare, så expanderar man användartillägngligheten på webbsidan. Kunder, samt kunder till kunder på Askås I&R utnyttjar olika webbläsare, så testning bör till viss grad kunna utföras på olika webbläsare.
4. Det sista kriteriet som verktyget bör uppfylla är uppenbart, dock kritiskt. Verktyget bör specifikt behandla automatiska funktionella tester för webbaserade system, såsom webbsidor.

2.4.2 Genomgång av verktyg

Utifrån de kriterier som beskrivs under 2.4.1; *Kriterier för verktyg*, så utförs en översiktlig undersökning av vilka verktyg som är relevanta på nätet. En källa som ger en lista på många verktyg som utför en mängd olika typer av testning, är Softwaregatest.com (2017). Källan listar näst intill all de kända verktygen som används för testning på webben i dagsläget. Allt från Load & Performance testing och Web Site Security testing tools till Web Functional testing tools. Sidan nämner också verktyg som Watir, och beskriver att det är ett verktyg som bygger sin funktionalitet på Selenium Webdriver. Webbplatsen beskriver ett 30 tal olika verktyg som används för testning på webben, som antingen bygger sin funktionalitet på, eller kan integrera med, verktyget Selenium. Andra verktyg som webbplatsen nämner är Sauce Labs, vilket är ett molnbaserat verktyg som kan utföra VM (Virtuell Maskin) tester på hundratals olika maskiner. Sidan nämner också javascript baserade verktyg som PhantomJS, som effektivt utför så kallade ‘huvudlösa’ tester, vilket betyder att de körs effektivt i bakgrunden på lokal maskin, genom att ej utnyttja en webbläsares miljö.

Webbsidan Testingtools.com (2017) har publicerat en lista för passande verktyg som används för att utföra funktionella tester. Här nämns ännu en gång Selenium som “Selenium is a popular automated web testing tool and helps you automate web browsers across different platforms.” (Testingtools.com, 2017), vilket översätter till “*Ett populärt automatiserat web testnings verktyg, som hjälper dig automatisera webbläsare över olika plattformar”*. Testingtools.com beskriver också att “Selenium has the support of some of the largest browser vendors who have taken steps to make Selenium a native part of their browser.” (Testingtools.com, 2017), vilket översätter till “*Selenium har stöd från några av de största webbläsare som har tagit steg för att göra Selenium en ursprunglig del av sin webbläsare*”. Testingtools.com ger också alternativ på många andra verktyg som är open-source, såsom Watir och Windmill, som bygger på programspråken Ruby, Python och Javascript.

Företaget Tieto sysselsätter omkring 14 000 anställda inom Norden (Tieto.se, 2017) och beskriver en tillgänglig tjänst som Frontend-utvecklare. Tjänsten innefattar att man jobbar med mjukvaruutveckling samt testning av mjukvara, och beskriver kunskaper inom verktyget och språket Selenium som meriterande. Vid vidare undersökning angående funktionstestning på Tieto (2017), så visade det sig att de bedriver funktionstester med både kostsamma ledande verktyg från HP och IBM, verktyg av lägre kostnad som TestComplete, samt open-source verktyg som Selenium.

Webbsidan Csjobb.idg.se (2017) anonserar också ut ett jobb som testautomatiserare på företaget Sogeti, som är ett av Sveriges största IT-företag. Tjänsten baserar på automatiserade tester, och personal eftertraktas som besitter erfarenheter i teknologier såsom QTP, Selenium, Ranorex och Protractor.

Vid vidare undersökning om Selenium så analyserar Holmes & Kellogg (2006) verktyget och dess funktionalitet. Verktyget sammanfattas som att hantera problem angående webbaserade tester väl, utan att tillägga fler problem i form av verktygsproblem. Verktyget beskrivs också som kraftfullt system med en användarkrets som växer kraftigt.

3. Implementation & metodik

Detta kapitel inleds med metodik för insamling av data från anställda på Askås I&R. Den insamlade datan används för att utvärdera vilket programspråk som bäst passar företaget vid användning av verktyget som implementeras senare i studien. Utöver detta så presenteras det verktyg och programspråk som användes för att tillämpa automatiska tester samt reflektioner kring varför just dessa valdes för uppgiften.

Utifrån den implementering som genomförs i denna uppsats, så måste det utföras många praktiska tillämpningar (testfall) för att samla in någon form av relevant data. För att uppnå någon form av resultat, med hänsyn till uppsatsen, så beskrivs även mer praktiska aspekter av utförandet i detta kapitel. Utföranden som kategoriseras som metod till denna undersökning, samt som strategi för implementering av de automatiska testerna, innefattar bland annat vilket verktyg som används, valet av programspråk (se 5.2 Analys - *Val av programspråk*).

Slutligen så utgår denna studie ur ett iterativt arbetssätt, genom att först undersöka potentiella programspråk-kandidater för att göra någon sorts utvärdering av vilket språk Askås I&R bör använda sig av vid implementation av automatiska funktionstester. På grund av detta så inleds metodredovisningen nedan med en metod av att samla in data angående tidigare erfarenheter av programspråk, för att sedan utvärdera datan genom en analys och slutsats. Efter detta utförs nästa steg, att genom en metod implementera testerna med hjälp av det valda programspråket.

3.1 Datainsamling

3.1.1 Öppen undersökningsenkät

En öppen undersökningsenkät blev utdelad till de anställda som utför tester på Askås I&R. Syftet med denna enkät är att få en förståelse av hur testning ser ut i dagsläget, hur ofta det testas, samt diverse programspråk-erfarenheter.

Frågor som ställdes i enkäten med motivering för varför de ingick i studien beskrivs nedan.

Personliga frågor:

[1]	Namn. (Lämnas tom om anonymitet föredras).
Motivering:	Namnet användes endast tidigt under studiens gång för att identifiera deltagarna. Namnet på deltagarna presenteras aldrig i uppsatsen, deltagarna får istället en benämning som refererar till deras individuella svar.
[2]	Vad är din nuvarande position på Askås I&R bortsett från att utföra tester? (kan även lämnas tom om anonymitet föredras).
Motivering:	Denna fråga var relevant då detta relateras till vilka programspråkserfarenheter som deltagaren besitter på företaget.

Frågor angående testning:

[T1] Hur mycket tid (generellt) lägger just du på test av systemet i veckan/ ny systemrelease?

Motivering: Denna fråga var relevant då det ger en uppfattning av hur mycket testning som utförs på företaget i dagsläget.

[T2] Tror du att Askås I&R borde utveckla sina testrutiner?

Motivering: Denna fråga var intressant då effektiviteten av testningsrutinera utvärderas av de anställda.

[T3] Hur tror du att testnings rutinerna kan effektiviseras?

Motivering: Detta var relevant då det visar hur insatta de anställda är av automation av testfall.

[T4] Borde Askås I&R spendera mer eller mindre resurser (tid) på att testa sin produkt?

Motivering: Denna fråga var relevant då man får en inblick i hur testningsrutinerna ses av de anställda.

Frågor angående programspråk och erfarenheter:

[P1] Har du tidigare programspråk-erfarenheter bortsett från Perl? Bocka i den erfarenhet du har kring respektive språk.

Motivering: Intressant då erfarenheter presenteras.

[P2] Har du erfarenhet av andra programspråk, i så fall vilket/ vilka?

Motivering: Kan vara relevant om många deltagare visar erfarenheter för ett alternativt programspråk.

[P3] Känner du att språket Perl är ett vitalt språk för dig när du utför testprotokoll?

Motivering: Relevant då byte av programspråk kan vara orealistiskt då programspråket är för integrerat i testningsrutinerna.

[P4] Skulle du kunna tänka dig att lära ett nytt programspråk om det innebar mer effektiv testning?

Motivering: Intressant att se om positivt initiativ finns för nytt potentiellt programspråk.

En öppen enkät valdes för att tillåta en större grad av diskussion, samtidigt som de anställda kan svara på enkäten i mån av tid. Enkäten presenterades som en webbenkät (se bilagan); systemarkitekten skickade ut en länk till de åtta personer som liksom han själv sysslar med testning.

Genom att undersöka vilket programspråk som de anställda generellt besitter störst tidigare kunskap av, får uppsatsen ett programspråk att utgå ifrån vid uppbyggnad av automatiska testfall. Data insamlat från denna enkät tillåter också diskussion om hur testning på företaget kan förbättras från de anställdas perspektiv. Resultat från enkät finns under 4.1 *Resultat från öppen enkät*.

3.1.2 Intervju med systemarkitekt på Askås I&R

En öppen intervju (Patel & Davidsson. 2014:82) utfördes med en systemarkitekt på Askås I&R angående hur testning struktureras i dagsläget. Intervjuns syfte är att få en helhetsbild

av hur testningsrutinera bemöts i dagsläget och hur de potentiellt kan förbättras. Resultat av intervjun finns under 4.2 *Intervjusvar från Systemarkitekt på Askås I&R*. Denna intervju ägde rum då systemarkitekten på Askås I&R är den person som leder testningsrutinerna på företaget och även ansvarar för testprotokollen och dess utdelande till utvecklarna.

3.2 Verktyg för automatiserade tester

När jag påbörjade min studie så undersökte jag vilka verktyg som var mest populära. (se 2.4 Verktyg för automatisering). Förutsättningarna för verktyget var att det skulle hantera funktionella tester på webbsidor, utvecklat genom open-source, samt vara flexibelt när det kommer till vilket programspråk som används. Verktyget som bäst uppfyllde de kriterier som ställdes var Selenium. För resonemang kring val av verktyg, se; 5.1 *Analys - Val av verktyg;*

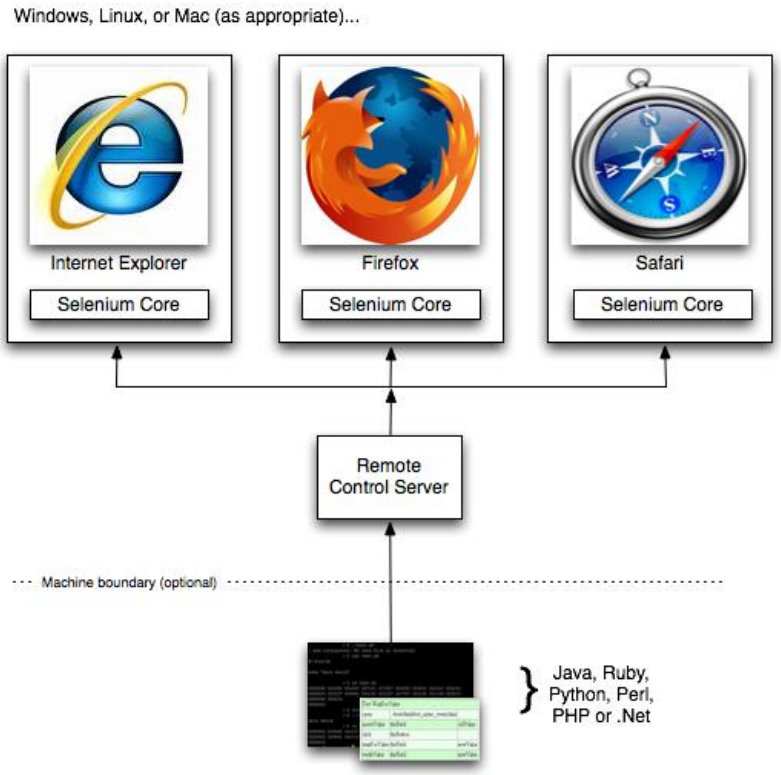
3.2.1 The Selenium Test Suite

The Selenium Test Suite, eller förkortat *Selenium*, är ett av de mest populära verktygen som används för att testa webbsidor idag. Selenium används för *browser automation*, vilket betyder att man automatiserar en separat webbläsare med hjälp av antigen en lokal Selenium Server (se 3.2.1.1 *Selenium Remote Control*), en automatiserad WebDriver (se 3.2.1.2 *Selenium WebDriver*), eller med hjälp av ett tillägg som finns tillgängligt för webbläsaren Mozilla Firefox (se 3.2.1.4 *Selenium IDE*). Selenium har också funktionaliteten att samtidigt utföra testfall på ett flertal servrar samtidigt med hjälp av Selenium Grid (se 3.2.1.3 *Selenium Grid*), detta gör att man kan utföra ett stort antal tester på mycket kort tid.

Selenium Remote Control, Selenium Webdriver, och Selenium Grid har alla funktionaliteten att utföra testfall på separata datorer på nätverket, genom användning av en *Selenium Standalone Server*. Detta tillåter användning av dedikerade datorer för testning, vilket ger möjlighet för en mängd kraftfulla testfall att köras samtidigt.
Selenium kommer också med separata bibliotek som ger språkstöd till de mest använda programspråk (Selenium HQ, 2017).
Selenium stödjer idag allmänna programspråk som Java, C#.net, Python, Ruby, och Javascript Node (Node JS).

3.2.1.1 Selenium Remote Control

Selenium Remote Control, även känd som *Selenium 1*, är den äldsta metoden som Selenium har utvecklat för automatiserad testning. Denna metod använder sig av en Selenium Server för att kunna öppna samt stänga en webbläsare, samtidigt som den fungerar som en HTTP proxy för att kunna ta emot requests från webbläsaren.
Testaren kan genom valfritt programspråk skriva sina testfall utifrån specifikationerna som är definierade, och sedan skicka dem till Selenium Servern på valfri maskin på nätverket. Servern kompilerar sedan testfallen och skickar dem till den lokala webbläsaren med hjälp av så kallade Selenium-Core commands, och på så sätt styra webbläsaren automatiskt (se figur 2).



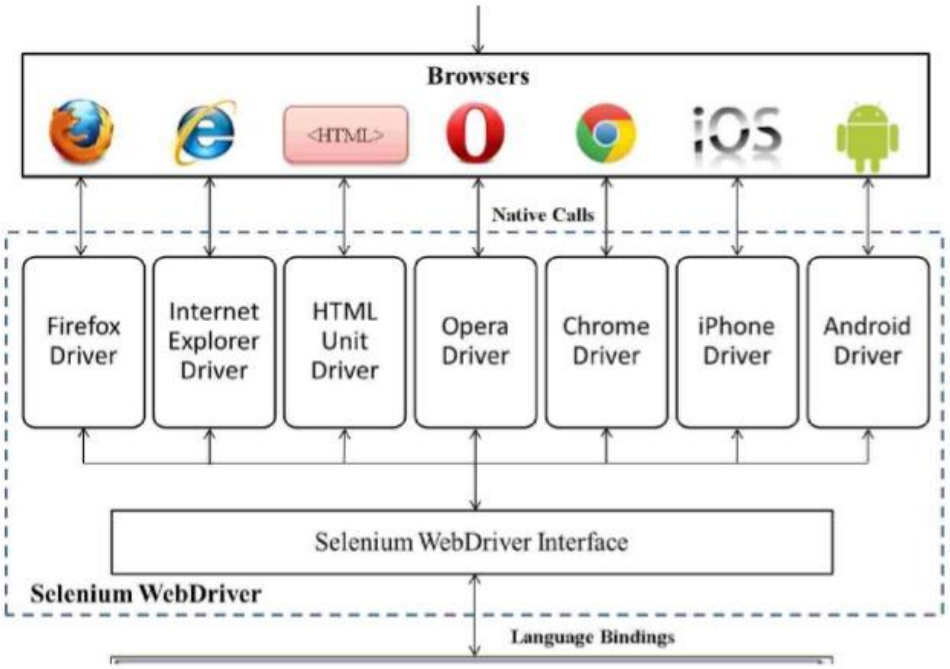
Figur 2. Illustration (Simplifierad) över hur Selenium Remote Control arbetar för att utföra tester. Hämtat från Selenium HQ [2017-03-15]

Notera att: Funktionaliteten som tillåter testaren att utföra testfall på en annan maskin är idag modernt utfört med hjälp av Selenium Grid (se 3.2.1.3 Selenium Grid).

3.2.1.2 Selenium WebDriver

Selenium WebDriver, även känd som *Selenium 2*, utgör det moderna sättet att automatisera testfall. Denna metod använder sig av en automatiserad förare, en så kallad WebDriver, för att kompilera testfall och sedan simulera användaren i vald webbläsare. Tidigare funktionalitet för webdrivers fanns tidigare utanför Selenium, då Selenium endast använde sig av Remote Control metoden (se 3.2.1.1). Selenium sammanslog sig sedan med webdriver för att undvika de begränsningar som Selenium Remote Control vanligtvis stod inför (Satish, Rahul & Dhanashree. 2015). Resultatet blev den moderna Selenium Webdriver.

Det finns en separat WebDriver för alla de webbläsare som används, några exempel på dessa är *Chrome Driver*, *Firefox Driver* och *Internet Explorer Driver* som då är automatiserade versioner av webbläsarna Google *Chrome*, Mozilla *Firefox* och Microsoft *Internet Explorer* (se figur 3).



Figur 3. Illustration över hur en WebDriver arbetar mot webbläsare. Hämtat från Software testing Studio [2017-03-15]

Man måste uppenbarligen också kunna styra dessa WebDrivers på något sätt, detta möjliggörs med något som kallas för *Language Bindings*. Dessa bindings är programspråk-bibliotek som utökar den syntax som finns för det programspråk som man har valt att utveckla testerna inom.

Notera att: WebDriver kräver att man använder de senaste versionerna av vald webbläsare, då denna metod använder sig av webbläsares inbyggda kompatibilitet för automatisering, vilket tillkom i de senare versionerna.

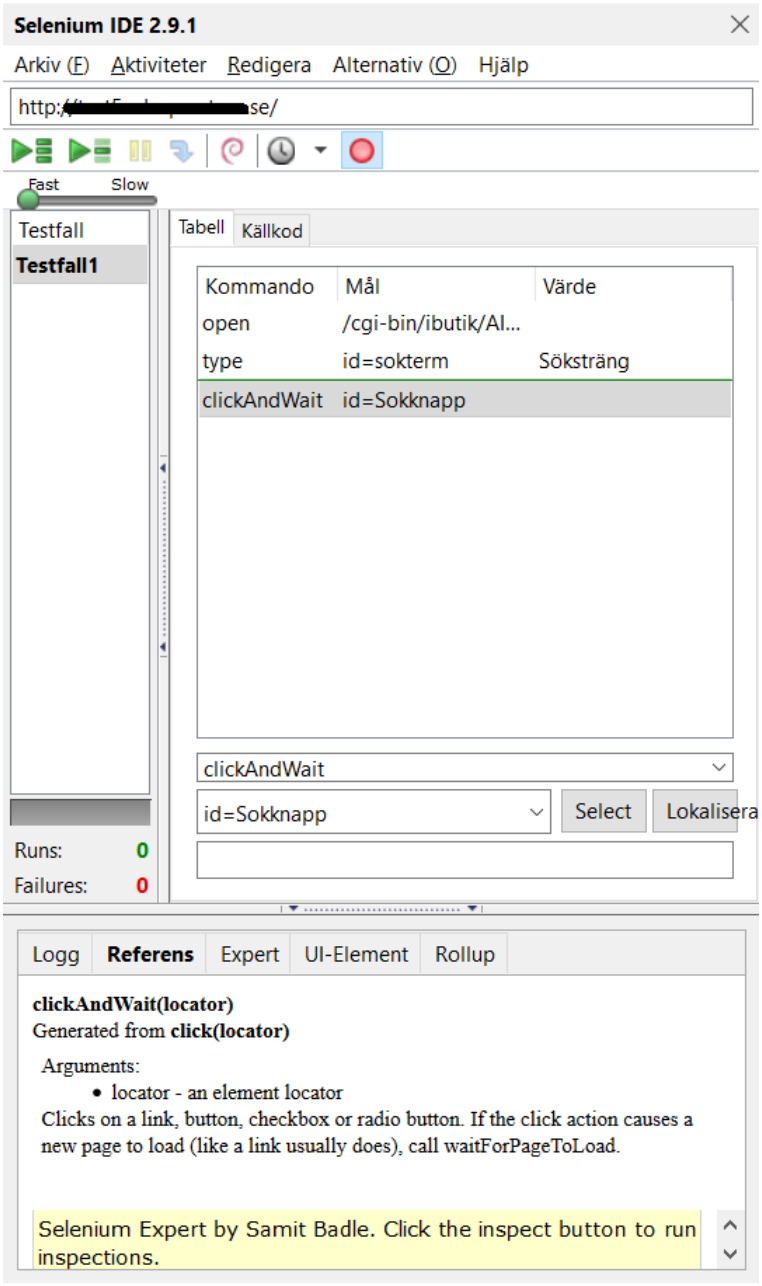
3.2.1.3 Selenium Grid

Selenium Grid är det effektiva och moderna sättet att utföra tester på andra maskiner genom nätverket. Implementeringen fungerar ungefär som vanlig användning av en Selenium WebDriver eller Selenium Remote Control, fast i stället för att utföra testet på en lokal webbläsare, så skickar Grid funktionaliteten ut kommandot genom en Selenium Standalone Server till en annan maskin på nätverket, som sedan kör testfallet genom sin lokala webbläsare. En Selenium Standalone Sever agerar som en centralpunkt på nätverket, och dirigerar ut testfall till valda maskiner.
Syftet med Selenium Grid är att kunna dedikera specifika maskiner som test-datorer på nätverket för att ge alla testfall samma förutsättningar.

3.2.1.4 Selenium IDE

Selenium IDE (Integrated Development Environment) är ett tillägg till de nyare versionerna av webbläsaren Mozilla Firefox som har funktionaliteten att kunna ‘spela in’ en användares aktivitet på en webbplats. Denna inspelning sparar man sedan undan och kan åkallas för att simulera en kopia av användaren, och sedan om något går fel så rapporteras felmeddelandet i verktyget (se figur 4).

Annan funktionalitet som är extra användbart med Selenium IDE är att testaren simpelt kan hitta/’track’ ett elements xpath-värde. Xpath-värdet är ett elements exakta plats på webbsidan. Då dessa värden kan vara mycket krångliga att lokalisera så kan man utnyttja Selenium IDEs funktionalitet för att enkelt hitta dem. Denna metod är mycket användbar medans man automatiserar med hjälp av andra verktyg.



Figur 4. Skärmklipp av Selenium IDE till Mozilla Firefox.

Notera att: Selenium IDE kan användas för att väldigt enkelt och effektivt genomföra översiktliga funktionella tester på en webbplats. Om effektivitet prioriteras och kunskap finns, så bör grundliga tester utföras med hjälp av någon av de övriga metoderna.
För enkel användning av detta tillägg så bör även Selenium IDE Button installeras som tillägg för att simplifiera användning av Selenium IDE.

3.3 Jämföra testprotokoll

3.3.1 Uppbyggnad

Testprotokollet som ligger i fokus under studiens gång är ett testprotokoll som används av Askås I&R i dagsläget. Testprotokollet utförs senare i uppsatsen ur synvinkel från en manuell testare, samt ur synvinkel från ett automatiserat testprotokoll. Relevant data som eftertraktas från dessa metoder är sådana som total testtid och diverse problem som uppstod under testprotokollet.

Skärmklipp som visas nedan består av en uppsättning av steg. Dessa steg representerar de aktiviteter som testaren måste utföra för att klara av testfallet. Testfallen avslutas också av ett förväntat resultat. Detta resultat är det förväntade utfallet från utfört testfall, och bör alltid uppfyllas för att försäkra testaren om att steg blivit utförda korrekt. Om inkorrekt utfall resulteras, så kan det tyda på att något av två olika scenarios har utförts. Antingen så har testaren felaktigt följt stegen i testfallet, och därav fått ett felaktigt resultat, eller så är funktionalitet hos CMS/ Webbshop felaktig.

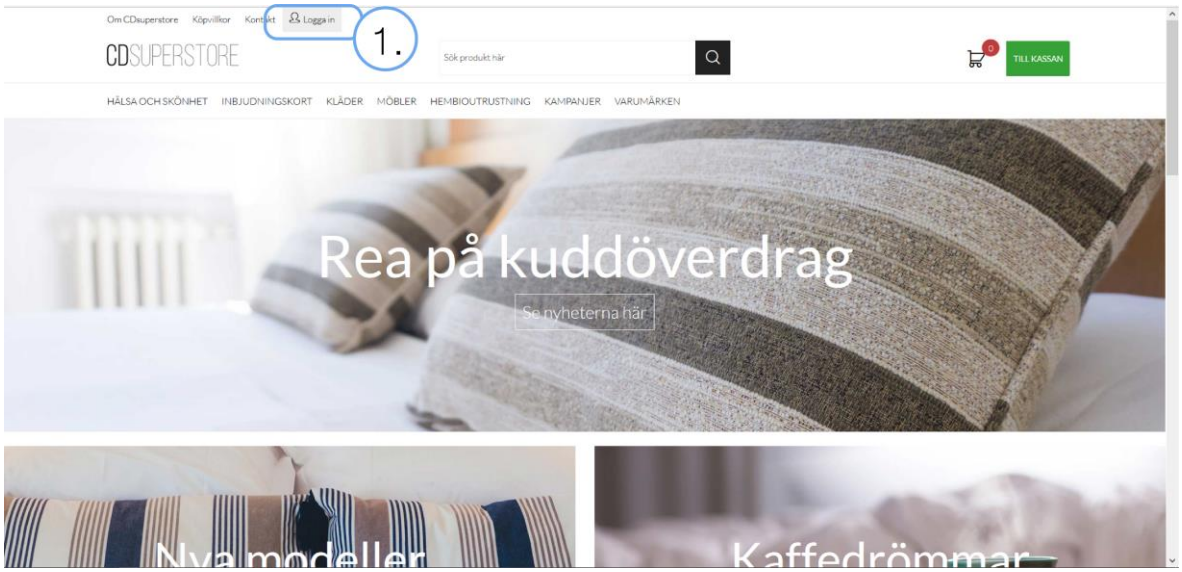
För enkelhetens skull så visas Testfall 1 och Testfall 2 i djup detalj, medan övriga testfall endast beskrivs översiktligt i studien. Två identiska testprotokoll kommer sedan att utföras, varav det första testfallet utgår från en manuell testare, och det senare utgår från ett automatiskt testprotokoll. Den manuella metoden härstammar från hur vissa testprotokoll utförs hos företaget idag, medan den automatiserade metoden är hur ett potentiellt fullt automatiserat testprotokoll kan se ut för företaget i framtiden. För att skilja på de olika testfallen från ett manuellt- och automatiskt perspektiv så benämns testfallen som ‘M’ för manuellt utfört, och ‘A’ för automatisk utfört testfall.

De testfall som sammanfattar de identiska testprotokollen presenteras nedan. För varje steg i de första två testfallen finns en korresponderande skämavbildning som namnges som ”Skärmklipp x.y”, där x står för testfallsnumret (alltså, antingen 1 eller 2) och y står för stegnumret.

- **[Testfall 1]** Skapa en ny kundprofil som användare.

Abstract
Syftet med detta testfall är att skapa en ny kundprofil på webbplatsen. Då detta är en funktion som ofta används av kunder, samt blivande kunder, så kan man se denna funktionalitet som viktig för webbplatsen. Skulle denna funktion vara felaktig så skulle alltså webbplatsen kunna förlora en större mängd potentiella kunder, vilket inte är att föredra.

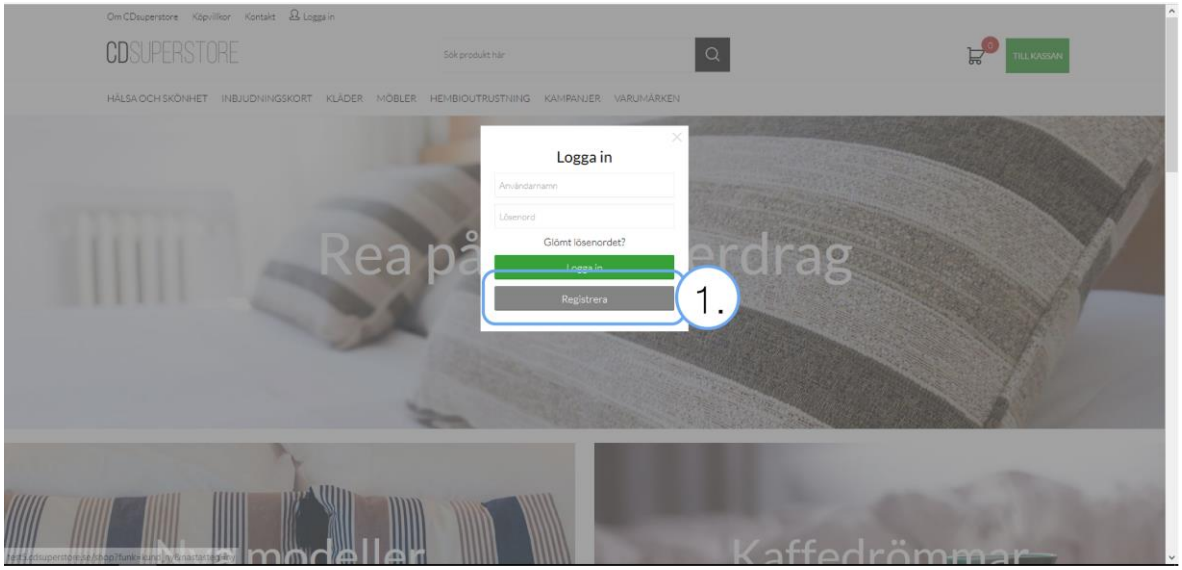
Steg 1. Navigera sig till att skapa ny användare via butikssidan.



Skärmklipp 1.1. Bild av huvudsidan av testshop.

1. Klicka på knappen ‘Logga in’.

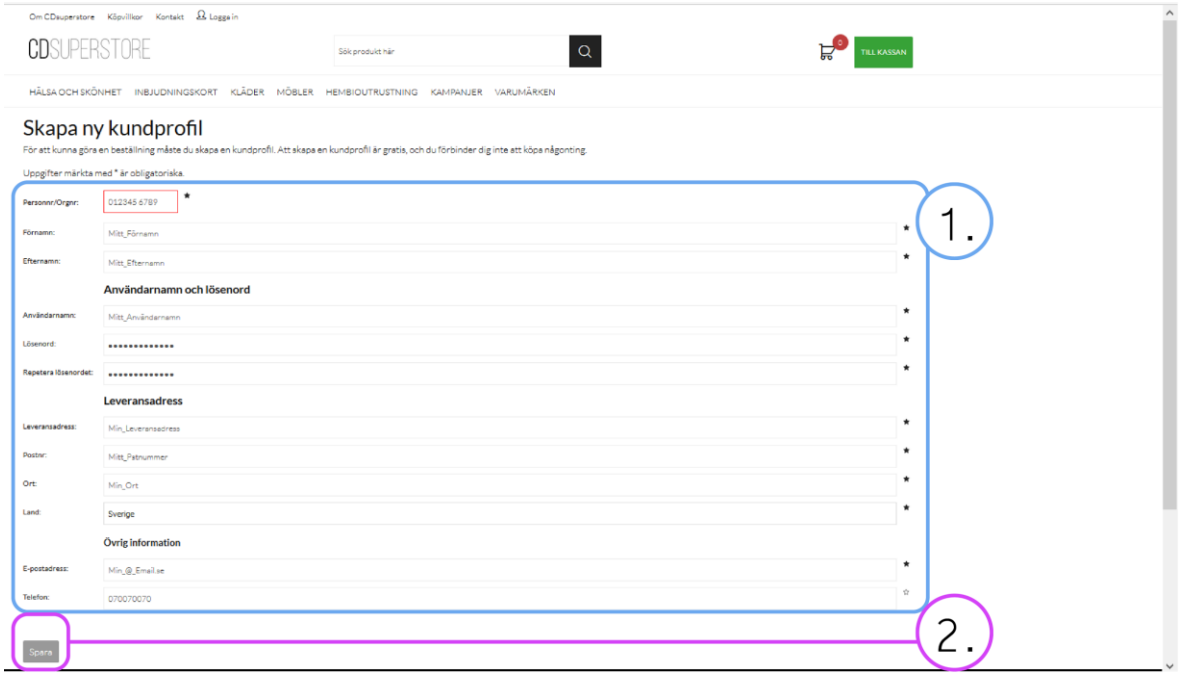
Steg 2. Navigera sig till att skapa ny användare.



Skärmklipp 1.2. Inloggning-/ Registrera fönster.

1. Klicka på knappen ‘Registrera’.

Steg 3. Fyll i kundformulär.



Skärmklipp 1.3. ‘Skapa ny kundprofil’.

1. Fyll i följande fält: *Personnr/Orgnr*, *Förnamn*, *Efternamn*, *Användarnamn*, *Lösenord*, *Repetera Lösenord*, *Leveransadress*, *Postnr*, *Ort*, *E-postadress* och *Telefon*. Fältet land skall vara automatiskt valt och kräver ej interaktion.
   För enkelhetens skull skall dessa värden användas:
   - Personnr/Orgnr: **0123 456789**
   - Förnamn: **Hans**
   - Efternamn: **Hult**
   - Användarnamn: **Hans0123**
   - Lösenord: **Hult6789**
   - Repetera lösenord: **Hult6789**
   - Leveransadress: **Gatan 25**
   - Postnr: **656 56**
   - Ort: **Karlstad**
   - E-postadress: **Hans01@mail.se**
   - Telefon: **012 345 6789**

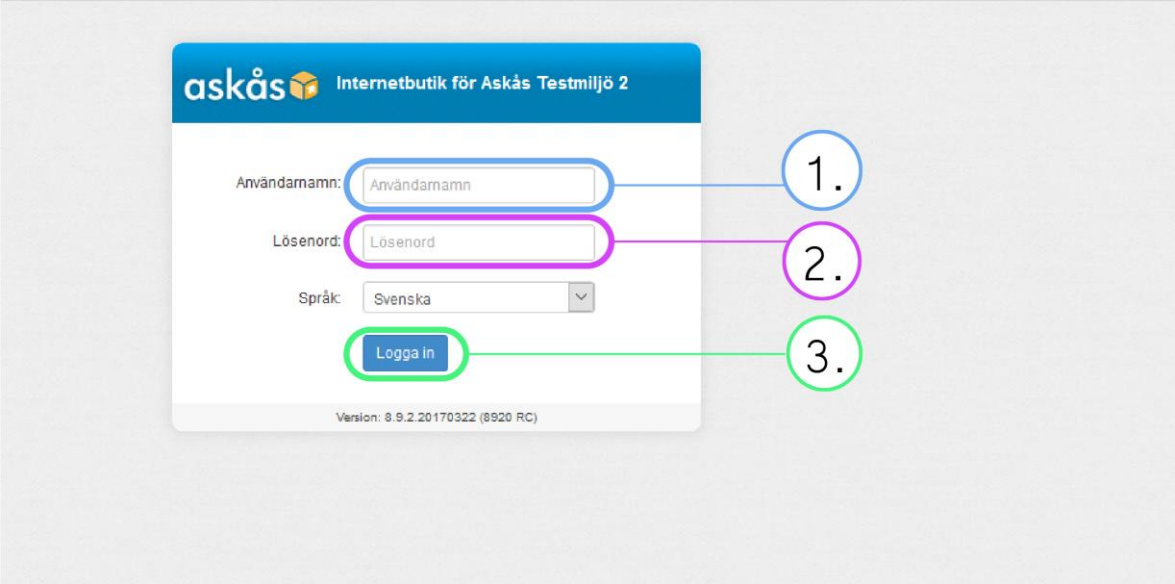
2. Testaren skall därefter klicka på knappen ‘Spara’ för att slutföra kundregistrering.

Förväntat utfall för Testfall 1:
Webbshop visas för användaren och inget specifikt fel uppstår på skärmen.

- **[Testfall 2]** Verifiera att profil finns via administrationssidan.

Abstract
Följt av att testaren har skapat en ny kundprofil på webbplatsen, så är det viktigt att verifiera att denna kundprofil har blivit inlagd i databasen. Detta verifierar testaren genom att leta upp skapad kundprofil genom administratörssidan, även kallad CMS.

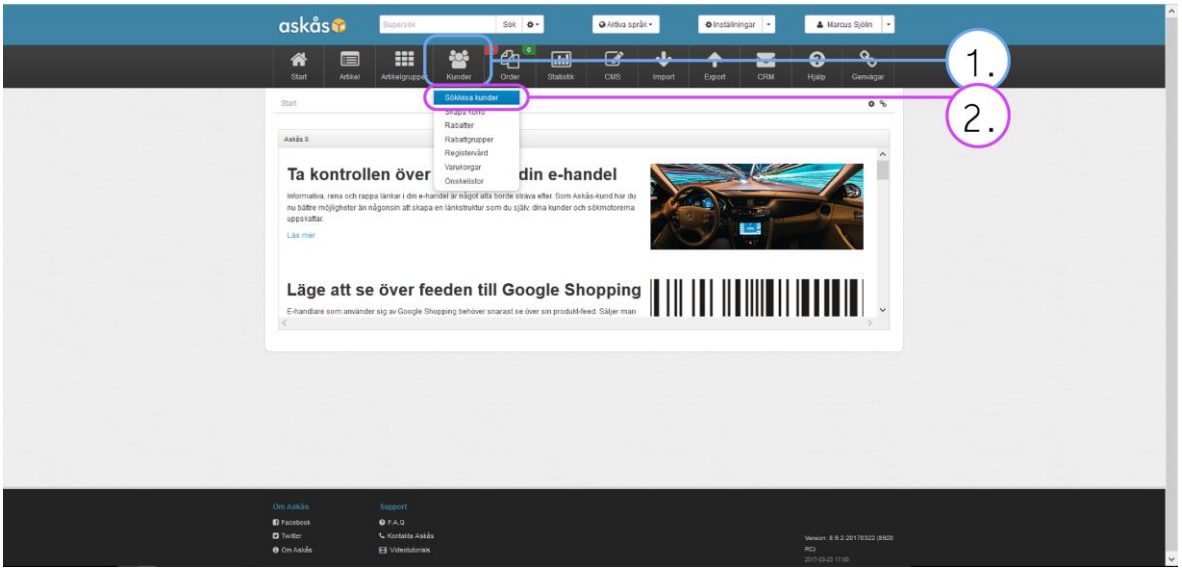
Steg 1. Logga in som administratör.



Skärmklipp 2.1. Inloggning på administratörssidan.

1. Fyll i användarnamn i fältet ‘Användarnamn’.
2. Fyll i lösenord i fältet ‘Lösenord’.
3. Klicka på knappen ‘Logga in’.

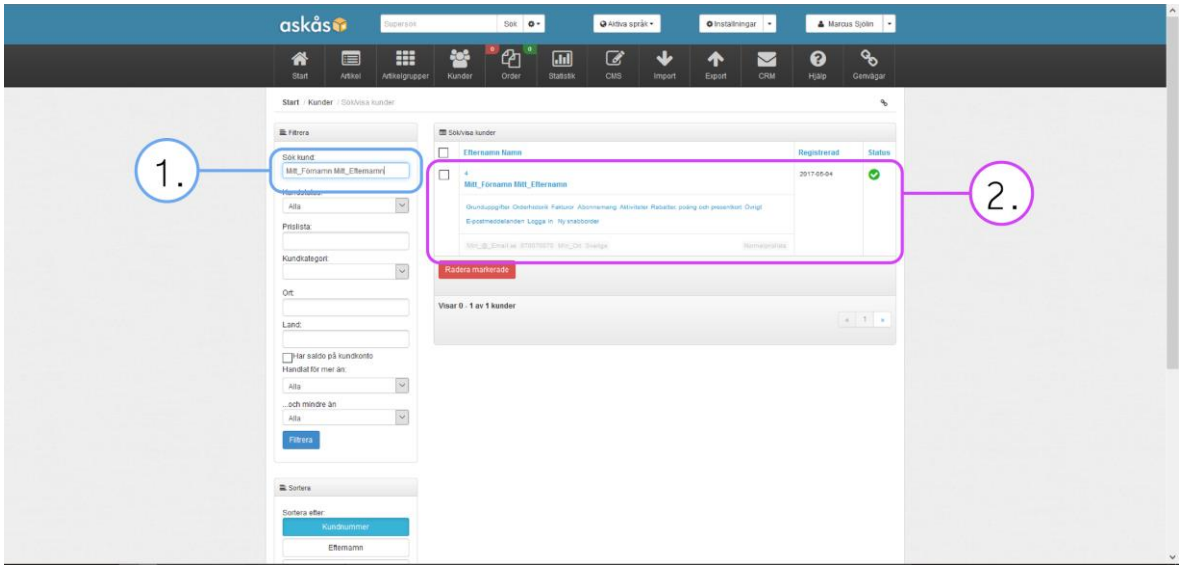
Steg 2. Navigera till kundlista.



Skärmklipp 2.2. Administratörssida med fokus Sök/visa kunder.

1. Klicka på navigationsknappen ‘Kunder’.
2. Klicka sedan på undermeny-objektet ‘Sök/visa kundlista’.

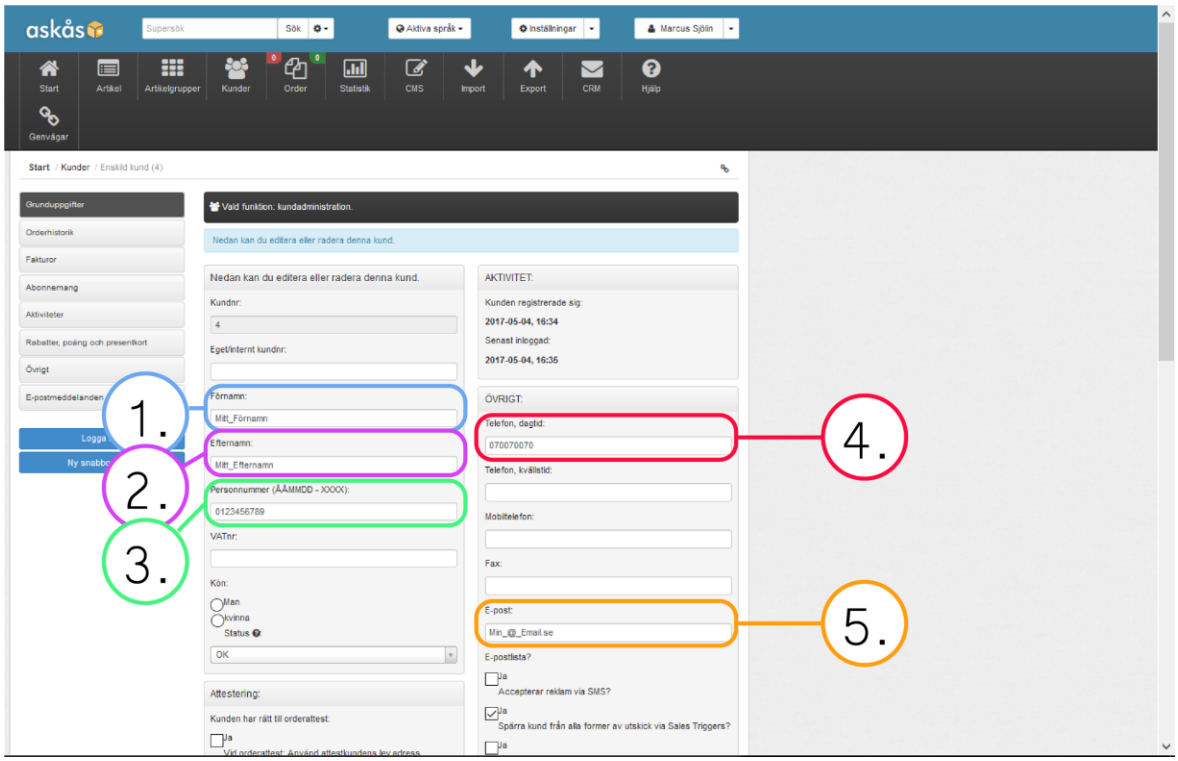
Steg 3. Sök fram kundprofil.



Skärmklipp 2.3. Sök/visa kundlista.

1. Använd sökfältet till vänster för att söka efter **Hans Hult**. Sökt kundprofil bör dyka upp under listan ‘Sök/visa kunder’. Klicka sedan på profilens förnamn eller efternamn för att navigera dig till vald kundprofil.

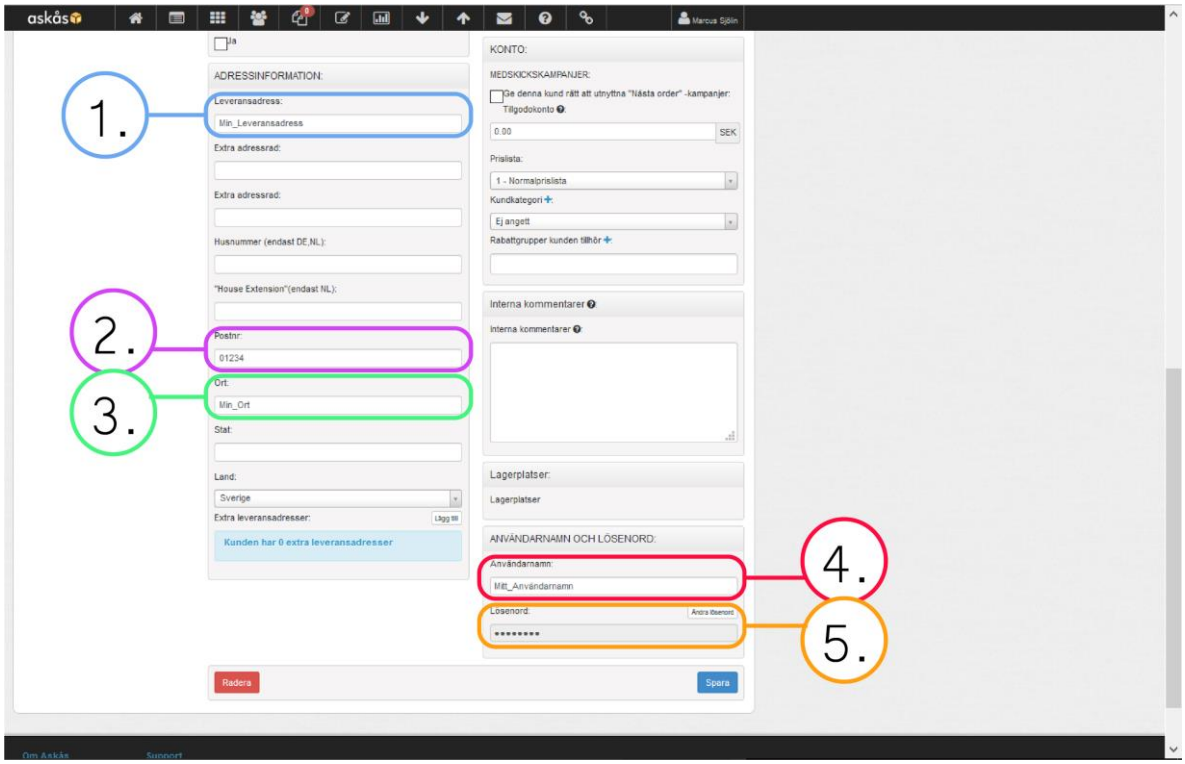
Steg 4 (1). Verifiera kundinformation hos kundprofil.



Skärmklipp 2.4(1). Kundprofil (1).

1. Verifiera att fältet ‘Förnamn’ innehåller värdet **Hans**.
2. Verifiera att fältet ‘Efternamn’ innehåller värdet **Hult**.
3. Verifiera att fältet ‘Personnummer’ innehåller värdet **0123456789**.
4. Verifiera att fältet ‘Telefon, dagtid’ innehåller värdet **012 345 6789**.
5. Verifiera att fältet ‘E-post’ innehåller värdet **Hans01@mail.se**.

Steg 4 (2). Verifiera kundinformation hos kundprofil.



Skärmklipp 2.4(2). Kundprofil (2).

1. Verifiera att fältet ‘Leveransadress’ innehåller värdet **Gatan 25**.
2. Verifiera att fältet ‘Postnr’ innehåller värdet **656 56**.
3. Verifiera att fältet ‘Ort’ innehåller värdet **Karlstad**.
4. Verifiera att fältet ‘Användarnamn’ innehåller värdet **Hans0123**.
5. Verifiera att fältet ‘Lösenord’ innehåller ett värde. Då dessa täcken är dolda, så får testaren istället undersöka om antalet tecken överensstämmer med lösenordet. Kolla därav att lösenordet innehåller **åtta tecken**, från **Hult6789**.

Förväntat resultat för Testfall 2:
Alla de fält som verifieras i Steg 4 överensstämmer med den data som angavs under Steg 3 i Testfall 1.

- **[Testfall 3]** Skapa en produkt till webbshoppen från administrationssidan.

Abstract
Testfallets syfte är att testa om funktionalitet för att tillföra en artikel finns hos CMS. Även denna funktion är extremt viktig för kunder och bör därför testas så ofta som möjligt.
Steg och instruktion för testfallet beskrivs nedan.

1. Via administrationssidan, klicka på navigationsknappen ‘Artikel’.
2. Klicka sedan på undermeny-objektet ‘Skapa ny artikel’.
3. Fyll först i huvudfältet ‘Artikelnummer’ med värdet ‘Artikel_1’ och klicka på ‘Verifiera och fortsätt’.
4. Fyll sedan i/ ändra de relevanta fälten nedan.
    - I fältet ‘Artikelns namn’, ange värdet ‘Namn_1’.
    - I fältet ‘Pris’, ange värdet ‘100’.
    - Ändra status för fältet ‘Artikel makulerad’* till värdet ‘Nej’.
5. Klicka på knappen ‘Spara’, som är lokaliserad längst ner till höger på sidan.

*’Artikel makulerad’ används om man vill skapa en produkt eller behålla en produkt i systemet utan att visa den i webbshoppen. Genom att ändra status på makulerad till ‘Nej’, så väljer man att framföra artikeln i webbshoppen.

Förväntat resultat för Testfall 3:
Skärmen styr testaren/ användaren längst upp på sidan. Där visas att ‘Grundinformation’ är ifyllt genom en mätare med färgen orange.

- **[Testfall 4]** Verifiera att artikel finns i webbshoppen som användare.

Abstract
Detta testfall baseras på Testfall 3 och används för att verifiera att korrekt artikel från föregående testfall framförs i butiken.

1. Via butikssidan, använd sökfältet och ange ‘Namn_1’ som sökvärde. Klicka sedan på sök knappen till höger.
2. Klicka på produkten nedan på sidan som besitter namnet ‘Namn_1’.
3. På produktsidan för produkten, verifiera att priset överensstämmer med definierat värde ‘100’ kr, och att artikelnumret överensstämmer med värdet ‘Artikel_1’.

Förväntat resultat för Testfall 4:
Korret artikel visas och innehåller korrekt pris. Denna produkt överensstämmer med den artikel som blev tillagd under Testfall 3.

- **[Testfall 5]** Lägg till den skapade produkten i varukorgen som användare.

Abstract
Via butikssidan, välj en valfri artikel inom webbshoppen. För enkelhetens skull, välj artikeln som skapades i Testfall 3. Lägg till denna artikel i varukorgen.

1. Använd sökfältet på startsidan för webbshop, och sök fram artikeln med namnet ‘Namn_1’, och gå sedan in på produktsidan för artikeln.
2. Klicka på den gröna knappen ‘LÄGG I KUNDVAGNEN’ till höger.

Förväntat resultat för Testfall 5:
Varukorgen blir fokuserad uppe till höger och visar att artikeln med korrekt namn blivit tillagd i varukorg/ kundvagn.

Ovanstående fem testfall är de som redovisas och diskuteras i denna uppsats. Sammanlagt använder sig företaget för närvarande av arton testfall.

3.3.2 Manuellt utförande

Detta avsnitt återspeglar hur testningen sker hos företaget idag.

Varje testfall beskrivs ur en manuell testares synvinkel och data resulteras i form av total testtid samt diverse svårigheter i testprotokollet, i form av kommentarer. Tid mäts från och med att startsidan har laddats, till och med att sista steget har utförts hos testfallet. Vid denna metod så används en testare som besitter vana vid testning av företagets CMS. Detta på grund av att det realistiskt är en erfaren testare som utför de manuella testfallen, och det är då tidsskillnaden mellan en testare med vana, samt de automatiserade testfallen som är av intresse.

Vid manuellt utförande så används ett verktyg i form av en tidtagarur på separat maskin. Resultatet redovisas nedan i avsnitt 4.3, *Resultat från manuellt utfört testprotokoll*.

3.3.3 Automatiserat utförande

När det gäller tidmätningen av den automatiserade testningen så utgör tillvägagångssättet bara en bild av hur testningen potentiellt kan se ut hos företaget.

Det automatiska testfallet utförs och tidtagning används för att få någon form av resultat. Tiden mäts från och med att startsidan har laddats, till och med att sista steget har utförts hos testfallet. Vid automatiskt utförande så används funktionalitet i scriptspråket för att mäta passerad tid. För resultat, se 4.4, *Resultat från automatiskt utfört testprotokoll*.

För att utforma de automatiserade testfallen så har studien främst utgått från ett elektroniskt bibliotek som kallas ‘WebdriverIO’ (WebdriverIO. 2017). Utöver WebdriverIO så bygger även

funktionalitet på kodexempel från *Selenium testing Tools Cookbook* (2017). Se avsnitt 3.2.1 *The Selenium Test suite* för hur verktyget används för att utforma automatiationen.

3.4 Etiska överväganden

Möjlighet gavs till respondenterna i de intervjuer som utförts under undersökningen att förbli anonyma i studiens sammanhang. Respondenterna gavs också rätten att neka till ljudinspelning och blev även informerade om att inspelningen endast kommer behandlas av studiens författare. Rubin & Chisnell (2008) skriver att respondenterna skall bli informerade om att inspelad ljudfil kommer att tas bort vid avslutad studie, vilket respondenterna blev.

De deltagare som deltog i enkätundersökningen på Askås I&R fick fördelen att delta i undersökningen vid valfri tidpunkt under en tidsperiod på två veckor. Deltagarna fick, likt respondenterna för intervjuer, möjligheten att förbli anonyma vad gäller både namn och position på företaget. Patel och Davidson (2008) beskriver att då en undersökning kan vara intresserad av information som kan tyckas vara känslig, så är det viktigt att deltagarna känner sig säkra i sin anonymitet, vid uttalande av sådan information. Det är viktigt att deltagarna skall ha möjligheten att hålla sin privata information eller företagsinformation anonym.

4. Resultat

4.1 Resultat från öppen enkät

En öppen undersökningsenkät blev utdelad till anställda som testar på Askås I&R (*se Bilaga*). Som beskrevs i Kapitel 3, var syftet med denna enkät att få en förståelse av hur testning ser ut i dagsläget, hur ofta det testas, samt diverse programspråk erfarenheter

Fem av åtta personer från företaget har medverkat på enkäten. Enkätdata beskrivs nedan från *Tabell 1* till och med *Tabell 6*, samt *Diagram 1*.

Tabell 1 - Persondata från enkätundersökning

Yrke/ Roll	Benämning
Webbutvecklare:	WU1
Webbutvecklare:	WU2
Webbutvecklare:	WU3
Utvecklare/ Systemarkitekt:	US1
Webbdesigner:	WD1

Frågeställning Testfrågor

[T1] - Hur mycket tid (generellt) lägger just du på test av system i veckan/ ny system release?
[T2] - Tycker du att Askås I&R borde utveckla sina testrutiner?
[T3] - Hur tror du att testrutinerna kan effektiviseras?
[T4] - Borde Askås I&R spendera mer eller mindre resurser (tid) på att testa sin produkt?

Tabell 2 - Svar **[T1]** - Hur mycket tid (generellt) lägger just du på test av system i veckan/ ny system release?

Person	Svar
WU1:	5h~
WU2:	2-4h
WU3:	3 timmar
US1:	Vid ny release, de få gånger jag har tid, så brukar bli ett par-tre timmar.
WD1:	Brukar bli upp mot 2 timmar i snitt per testomgång. Detta en gång varannan månad.

Tabell 3 - Svar **[T2]** - Tycker du att Askås I&R borde utveckla sina testrutiner?

Person	Svar
WU1:	Ja
WU2:	Det finns alltid rum för förbättring
WU3:	Ja.
US1:	Utan tvekan
WD1:	Ja, det vore bra ifall vissa moment gick att få mer automatiserade. Om man kör scriptet som går igenom t.ex. flödet i kassan och loggar hur processen går. Det är lätt att missa att testa alla scenarion.

Tabell 4 - Svar **[T3]** - Hur tror du att testrutinerna kan effektiviseras?

Person	Svar [T3]
WU1:	Mer automatisering av tester, inkl. modernare rutiner när det kommer till building, staging och deploying.
WU2:	Mer automation och standardiserade tester och lättare testcase
WU3:	Vi borde automatisera till högre grad. Tyvärr sätter kodens struktur i många fall stopp för detta. Dessutom tar det tid att skriva tester. Tid vi inte riktigt har just nu. Så vi sitter fast i nåt slags moment där.
US1:	Alla uppgifter över ca. 10h bör ha minst ett flödesdiagram med såklart så mycket dokumentation som möjligt, som i sin enkelhet förklarar exakt hur ny funktionalitet ska fungera. Detta gör det inte bara enklare att verifier om funktionen beter sig som den ska. Det är även ett stort plus att ha vid underhåll/vidareutveckling i senare skede. Jag tror även att vi behöver skifta fokus på hur vi testar saker. I dagsläget testar vi för mycket manuellt, vilket medför en hel del risker. Vårt fokus borde ligga på att automatisera testningen och att genomföra justeringar av systemet för att tillåta automatiserad testning till en högre grad. Vi borde även ha en person vars enda arbetsuppgift, dvs på heltid, är att förbättra testningsprocessen och höja kvalitén på produkten. Som en del i testprocessen borde vi även genomföra kvalitétkontroller vid jämna mellanrum, dvs kontrollera att vi arbetar på ett konsekvent och logiskt sätt med informationen (fraser) och funktionaliteten som vi introducerar i webbadmin.
WD1:	Lite enligt ovan fråga 2, samt om det nyutvecklade är lite mer uppsatt av utföraren i butik innan testaren testar.

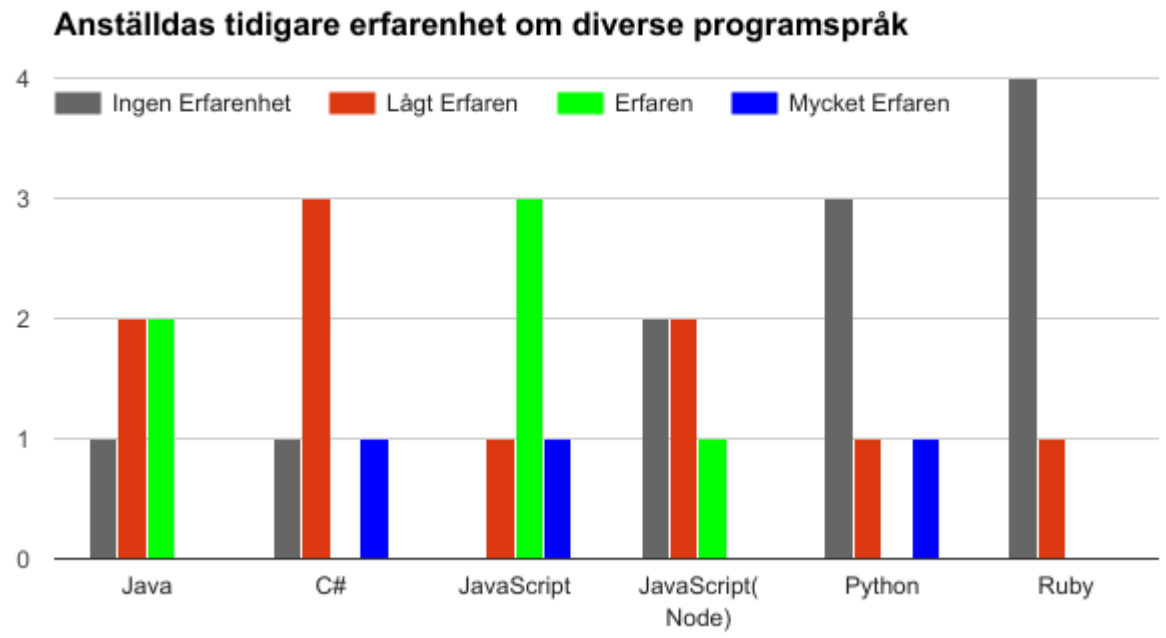
Tabell 5 - Svar **[T4]** - Borde Askås I&R spendera mer eller mindre resurser (tid) på att testa sin produkt?

Person	Svar [T4]
WU1:	Ja
WU2:	Mer eller samma tid som nu
WU3:	Mindre.
US1:	Mycket mer.
WD1:	Mer tid, mer automatiserat. Det finns en fördel att olika medarbetare får se nya funktioner och testa dessa.

Frågeställning Programspråkfrågor

[P1] - Har du tidigare programspråk-erfarenheter bortsett från Perl? Bocka i den erfarenhet du har kring respektive språk.
[P2] - Har du erfarenhet av andra programspråk, i så fall vilket/ vilka?
[P3] - Känner du att språket Perl är ett vitalt språk för dig när du utför testprotokoll?
[P4] - Skulle du kunna tänka dig att lära dig ett nytt programspråk om det innebar mer effektiv testning?

Diagram 1 - Svar **[P1]** – De anställdas programspråkserfarenheter.



Tabell 6 - Svar **P2**, **P3**, **P4**.

Person	Svar [P2]	Svar [P3]	Svar [P4]
WU1	-	Nej.	Absolut
WU2	-	Nej.	Absolut
WU3	-	Vissa saker blir svårt att testa ordentligt utanför Perl, men allt som skrivs ut i butiken kvittar egentligen.	Ja, då testmodulerna i andra språk som JS & Python är mycket effektivare och enklare att jobba med. Speciellt när det kommer till simulering av användarupplevelsen.
US1	C++, PHP	Lättare att korrigera eventuella fel själv om det är i Perl	Ja.
WD1	Inget.	Oklart.	Oklart.

4.2 Intervjusvar från Systemarkitekt på Askås I&R

Dessa frågor med tillhörande svar (Tabell 9) är resultatet från intervjun som kort beskrivs i avsnitt 3.1.2 *Intervju med Systemarkitekt på Askås I&R*.

Fråga 1:
Hur ser företaget, och du som systemarkitekt på testning och hur arbetar ni med testning idag?

Fråga 2:
Hur arbetar ni med testning idag?

Fråga 3:
Kör ni bara Manuella tester idag?

Fråga 4:
Kan testnings-rutinerna förbättras? Uppföljning: Varför blir rutinerna ej förbättrade?

Fråga 5:
Hur ser optimal testning ut för företaget idag? (Hur vill du som Systemarkitekt att testrutinerna skall vara)?

Tabell 9 - Svar från Intervju med Systemarkitekt på Askås I&R

Fråga	Svar
1:	Vi testar ganska ofta eftersom vi släpper nya versioner av systemet varannan vecka. På grund av det så testar vi också varannan vecka. Historiskt kanske testning varit mindre men med åren har testning blivit viktigare för företaget. Det är alltid en fråga om resurser och tid, det är mycket som ska passa in. Vi lägger mycket tid på testning, men vi kan göra det bättre. Tiden vi lägger på testning är rimlig, men vi behöver effektivisera vägen fram till testerna, vi behöver en bättre (tydligare) testspec i ett tidigare skede av utvecklingsfasen. Vi uppmanar att utvecklarna att göra en testprotokoll (som en intern spec för att specificera hur det ska bli, alltså resultatet som ska bli av utvecklingen) tidigt i uppstartsfasen.
2:	Börjar med ett 'bollplank': En utvecklare som är bollplank och en annan som är utvecklare/testare. Man diskuterar sedan också med 'bollplanket' och strukturerar upp testprotokollet i ett tidigt stadie. Sedan skapar utvecklaren/testaren, som ansvarar för testbar kod, testprotokollet. 'Bollplanket' utför sedan testfallet för att hitta eventuella slarvfel. Detta fungerar lite som ett internt test innan testprotokollet släpps som ett sluttest. Exempel: Det kan slinka igenom saker när man skapar testfallet som kan vara ett resultatet av stress. Det är viktigt att detta inte hamnar i sluttestet. Ett litet fel som skapats på grund av slarv eller dålig kommunikation kan resultera i ett framtida fel som tar ett flertal timmar att utreda.
3:	Vi kör en del automatiska tester, delvis via Selenium, på tester som vanligtvis utförs. Vi har även ett api som vi tester, som vi kör tester på (i sluttest) som även körs automatiskt. Vi försöker få folk att använda selenium men det är en tröskel eftersom det är något nytt att lära sig för utvecklarna och tiden finns inte riktigt. Många vill gå över. Konsekvensen blir att: Testprotokollet blir inte riktigt komplett då och kvalitén på koden kan även påverkas och det påverkar även sluttestet eftersom personen som testar slutprotokollet inte vet ifall utvecklaren testat allt innan sluttestet, det kan va saker som borde testas som utvecklarna inte tänker på. Många av de testfall som utförs vid ny systemrelease är också svåra att automatisera.
4:	De som testar hinner inte riktigt med testningen och då blir det jag (Systemarkitekten) som måste ta över testprotokollet. Detta gör att jag får väldigt mycket att testa under väldigt kort tid och detta kan ge konsekvenser som att man missar vissa fel. De som testar (utvecklare och några designers) är med i testprocessen men det är inte säkert att alla tänker på allt i testningsprocessen. Utöver detta så bör testrutiner förbättras i ett tidigare skede under utvecklingen.
5:	Genom att tidigt i utvecklingsprocessen speca hur testet ska se ut. Bli hårdare på kodgranskning, dels att man har ett antal automatiska testfall som täcker över stora delar av butiken. Att få automatiska testfall som automatiskt triggas vid uppdatering. Genom att ha en mängd automatiska testfall som körs som testar så att all grundlig funktionalitet alltid funkar vid ny systemversion, som att lägga till i varukorg, skapa produkt, samt skapa användare. En annan funktion som skulle vara bra att implementera vore att ha en testpanel för utvecklare där de lätt kan åkalla diverse testfall, för att effektivt kunna testa grundlig funktionalitet/ integration med systemet. Det finns dock problem, exempelvis när en butik konfigureras så kan det påverka en annan butik på ett helt annorlunda sätt. Detta på grund av att butikerna är konfigurerade på massvis olika sätt.

4.3 Resultat från manuellt utfört testprotokoll

Tabell 10 - Resultat från manuellt utfört testprotokoll.

Testfallets benämning:	Utförande nummer:	Tid för utförande: (sekunder)	Kommentar:	Medelvärde för utförande:
Testfall **M1**	1	**45,0**	Tidskrävande för testaren att lokalisera diverse fält.	**44,8 sekunder**
	2	**44,5**		
	3	**48,4**		
	4	**45,0**		
	5	**40,9**		
Testfall **M2**	1	**54,1**	Tidskrävande för testaren att lokalisera, samt jämföra diverse fält med fördefinierade värden.	**52,1 sekunder**
	2	**52,6**		
	3	**52,0**		
	4	**49,8**		
	5	**51,8**		
Testfall **M3**	1	**26,3**	Tidskrävande för testaren att lokalisera vart fält och definiera dess värde.	**24,2 sekunder**
	2	**22,3**		
	3	**26,0**		
	4	**24,6**		
	5	**21,5**		
Testfall **M4**	1	**9,2**	Testfall genomfördes snabbt och utan hinder.	**8,9 sekunder**
	2	**8,9**		
	3	**8,8**		
	4	**9,0**		
	5	**8,6**		
Testfall **M5**	1	**6,7**	Testfallet utfördes snabbt och utan hinder.	**6,6 sekunder**
	2	**6,4**		
	3	**7,4**		
	4	**6,5**		
	5	**6,0**		

4.4 Resultat från automatiserat testprotokoll

Tabell 11 - Resultat från automatiskt utfört testprotokoll.

Testfallets benämning:	Utförande nummer:	Tid för utförande: (sekunder)	Kommentar:	Medelvärde för utförande:
Testfall **A1**	1	**3,9**	Felfritt utförande.	**3,6 sekunder**
	2	**3,2**		
	3	**3,4**		
	4	**3,4**		
	5	**3,9**		
Testfall **A2**	1	**4,2**	Felfritt utförande.	**4,3 sekunder**
	2	**4,5**		
	3	**3,9**		
	4	**4,9**		
	5	**4,2**		
Testfall **A3**	1	**6,4**	Mycket olika utfall från vart utförande. Detta tros bero på lokal maskin eller nätverksfördröjning.	**7,2 sekunder**
	2	**8,0**		
	3	**6,5**		
	4	**7,5**		
	5	**7,7**		
Testfall **A4**	1	**2,2**	Felfritt utförande.	**2,5 sekunder**
	2	**3,3**		
	3	**3,4**		
	4	**1,2**		
	5	**2,3**		
Testfall **A5**	1	**1,2**	Felfritt utförande.	**1,6 sekunder**
	2	**1,3**		
	3	**2,5**		
	4	**2,3**		
	5	**2,2**		

5. Analys

Den analys som genomförts består av två separata delar. Den första delen inkluderar val av verktyg, vilket utfördes tidigt i studiens gång. Den andra delen inkluderar analys av den data som samlats in för att besvara de undersökningsfrågor som ställts i studien. Anledningen till detta upplägg är att verktygsvalet var vitalt tidigt i studien. På grund av detta så redovisas analysen som ledde fram till valet av verktyg, även om denna frågeställning ej behandlas som en undersökningsfråga. Val av verktyg behandlas i 5.1 medan 5.2-5.4 analyserar data rörande programspråk, testfallsprioritering respektive tidsskillnader.

5.1 Val av verktyg

Här analyseras den data som är relevant angående vilket verktyg som är av intresse. Denna analys baseras på data från 2.4 V*erktyg för automatisering*, samt 4.2 *Intervjusvar från Systemarkitekt på Askås I&R*. Ett verktyg som var mycket populärt enligt testingtools.com (2017), samt uppfyllde mina kriterier var Selenium. Verktyget uppfyller de kriterier som ställdes mot verktyget. De kriterierna som verktyget bör uppfylla var att det skulle vara tillgängligt utan kostnad av finansiella skäl, det bör vara flexibelt när det handlar om vilket programspråk som kan användas vid utveckling, verktyget borde också stödja majoriteten av moderna webbläsare då det höjer kvaliteten på testning för Askås I&R, samt att verktyget bör behandla automatiska funktionstester på webbaserade system.

Då verktyg såsom Watir, Windmill och PhantomJS beskrivs som kraftfulla verktyg av de källor som jag har utgått ifrån, så är dessa verktyg ytters begränsade. Delvis på grund av att dessa språk begränsas till ett specifikt programspråk, eller till ett fåtal webbläsare. Watir beskrivs som ett kraftfullt verktyg som är mycket effektivt vid automatiserade tester, dock baseras detta verktyg på programspråket Ruby. Detta gör att verktyget blir ointressant, då verktyget måste bygga på ett programspråk som passar företaget. PhantomJS kan då tyckas vara ett intressant verktyg, då det baseras på programspråket Javascript, vilket Askås I&R, som ett webbaserat företag, uppenbarligen har stor kunskap av. Dock, det som får huvudlösa verktyg som PhantomJS att förbli ointressanta, är just att de är huvudlösa. Ett huvudlöst verktyg existerar utan ett användargränssnitt, vilket får testerna att bedrivas genom en bakgrundsprocess. Detta känns för mig som ett för avancerat verktyg i ljuset av denna studies syfte. Som blivande testare så vill jag interagera med hur det automatiserade testfallet utförs, och därför valde jag att använda ett verktyg som körs genom en webbläsare med hjälp av en webdriver.

Av de verktyg som analyserades var det Selenium som framstod som mest relevant, då Selenium besitter stor funktionalitet som en mängd andra verktyg bygger sin funktionalitet på. Softwareqatest.com (2017) gav en lång lista på potentiella verktyg och beskrev Selenium som ett open-source verktyg som innehåller många mindre relaterade verktyg, samt att Selenium WebDriver har funktionaliteten att interagera med operativsystemet.

Selenium är ännu mer intressant på grundval av intervjun med systemarkitekten på företaget (4.2 *Intervjusvar från Systemarkitekt på Askås I&R*). Han förklarar att företaget redan har ett testprotokoll som besitter viss funktionalitet för automation genom Selenium. System-arkitekten förklarar att den funktionaliteten finns hos testfall som körs ofta vid ny system-version.

Utifrån det programspråk som valdes att utgå ifrån (se 5.2 *Val av programspråk*), så visade det sig att Selenium har stöd för programspråket javascript. I detta fall i form av Node.js. Utifrån detta framstår valet av verktyget Selenium som självklart. Verktyget uppfyller de kriterier som ställs mot verktyget. Askås I&R besitter också viss tidigare erfarenhet av verktyget.

5.2 Val av programspråk

Denna analys utgår från undersökningens enkät som gavs till de anställda på Askås I&R. Hälften av frågorna (P) ställdes för att få en förståelse av vilka programspråk som de anställda besitter kunskap om, hur väl programspråket Perl binder sig till de funktionella testerna, samt om de anställda kunde tänka sig att lära sig ett nytt programspråk om det innebar effektivare testning genom automatisering. Endast fem anställda deltog i undersökningen. Om alla åtta testare deltagit, så kunde det möjligen valet av programspråk blivit ett annat..

[P1] Analys utgår från *Diagram 1* (se 4.1 *Resultat från öppen enkät*), övriga enkätfrågor angående tidigare erfarenheter, samt det stöd som Selenium ger till programspråket. Utifrån Diagram 1 så utmärker sig Javascript som ett programspråk/ skriptspråk som de flesta anställda besitter tidigare kunskaper av. Det finns också viss kunskap för Programspråket Javascript (Node), vilket är positivt, då det baseras på Javascript. Detta gör att Javascript (Node), även kallat Node.js, kan vara relativt lätt att lära sig, då det byggs på Javascript. Utöver dessa språk så var det endast programspråket Java som ett antal anställda besatt en relativt bra färdighet utav. Generellt så visade det sig att de de anställda besatt minst kunskap utav C#, Python och Ruby.

[P2] En deltagare kommenterade också i Tabell 6 att hen besatt kunskaper inom programspråken C++ och PHP. Då detta endast blev kommenterat av en deltagare, så påverkar detta ej utfallet (se Tabell 6).

[P3] I studiens syfte så frågades det också om programspråket Pearls relation till utförande av testprotokoll. Denna fråga ställdes då det var intressant att veta om Perl är viktigt för testernas utförande. Om det hade visat sig vara ett vitalt språk för de flesta deltagare, så hade det kunna tyda på att det skulle vara ovist att byta programspråk. Resultatet från denna fråga var blandade (se Tabell 7). Dock utifrån de kommentarer som resulterades, så utgår Perl ej som ett vitalt språk.

[P4] Den slutliga frågan i enkäten var om den anställde kunde tänka sig att lära ett nytt programspråk om det innebar mer effektiv testning. Resultatet i Tabell 8 visar att de anställda som deltog i enkäten var positiva till idén. Denna fråga var relevant då resultatet tillät användningen av ett alternativt programspråk till studiens uppgift.

Utifrån denna genomgång av de svar från undersökning enkäten, så framstår javascript som det programspråk som flest anställda besitter störst kunskaper inom. Javascript kom att användas vid utveckling av de automatiserade testprotokollen. Detta resultat var förväntat, då Askås I&R är ett delvis webbfokuserat företag. Varje företag som bedriver

webbutveckling har en tendens att ha personal som besitter kunskap i programmering på webbnivå.

Notera att: Denna analys baseras på det resultat som endast fem anställda bidrog till genom en enkät på företaget. På grund av det låga deltagarantalet så kan resultatet potentiellt se annorlunda ut, vid en enkät med fler deltagare.

5.3 Testfallsprioritering

Blom förespråkar för att man skall automatisera de tester som upprepas (se 2.1 Öppen *intervju med Universitetslektor Martin Blom på Karlstads Universitet*). Askås I&R fokuserar också på att automatisera de testfall som vanligtvis utförs vid nya systemreleaser. Viss automation finns redan på företaget för testfall som upprepas ofta (se 4.2 *Intervjusvar från Systemarkitekt på Askås I&R*). Att prioritera automatisering till de testfall som upprepas oftast resulterar i störst tidsbesparing.

Att prioritera de testfall som oftast utförs är inte en av de nio olika prioriteringsstrategier som Rothermel et al. (1993) beskriver. En av prioriteringsstrategierna som däremot passar in på den testfallsprioritering som Blom och systemarkitekten på Askås I&R beskriver, är *Optimal prioritization*, vilket utgår från att prioritera de testfall som förväntas att hitta flest fel. Att prioritera automatisering av testfall som utförs ofta kan överensstämma med att prioritera testfall som med störst sannolikhet hittar fel, då de testfall som utförs oftast av Askås I&R prioriteras av en anledning. Dessa två prioriteringsstrategier har identiska mål, att prioritera och utföra de testfall som sannolikt resulterar i att hitta flest fel.

Askås I&R prioriterar automation av de testfall som testar central funktionalitet hos CMS och butikssida, som skapande av kunder, artiklar, samt betalningsfunktion. Dessa funktionaliteter kan ses som centrala och vitala för kunder. Att hitta ett fel hos dessa funktionaliteter prioriteras, då sådana fel kan påverka kunderna drastiskt. Av detta kan man se att Askås I&R prioriterar testfall som upptäcker vitala fel som kan påverka kunder på ett negativt sätt. Även om de testfall som Askås I&R prioriterar potentiellt kan upptäcka vitala fel, så kan de även hitta ett flertal mindre kvantitativa fel av den anledning av att funktionaliteten som testas är central i systemet. Central funktionalitet som förbinder stora delar av systemet kan resultera i ett flertal mindre fel hos nya systemreleaser. Med detta i åtanke bedriver Askås I&R i dagsläget en variant av *Optimal prioritization*, då dessa centrala funktioner testas.

Prioritering genom endast *Optimal prioritization* hos Askås I&R skulle resultera i att prioritera de testfall som med störst sannolikhet skulle hitta nya fel hos systemet. Till skillnad från prioriteringen av vitala fel som används i dagsläget, så skulle denna prioritering utgå från att hitta ett större antal fel före vitala fel.

5.4 Tidsskillnad

Vid jämförelse av tidskonsumtion hos testfallen så jämförs Tabell 10 och Tabell 11, för att urskilja proportionell potentiell effektivisering hos de identiska testprotokollen. Total tid för båda testprotokoll sammanställs sedan och en proportionell tidsskillnad presenteras.

Tabell 13 - Sammanställda resultat från testprotokoll.

Testfallets benämning:	Tid - Manuellt testfall:	Tid - Automatiserade testfall:	Tidsskillnad
Testfall **1**	**44,8** sekunder	**3,6** sekunder	Upptar **8,0%** av manuell tid.
Testfall **2**	**52,1** sekunder	**4,2** sekunder	Upptar **8,0%** av manuell tid.
Testfall **3**	**24,2** sekunder	**7,2** sekunder	Upptar **29,8%** av manuell tid.
Testfall **4**	**8,9** sekunder	**2,5** sekunder	Upptar **28,1%** av manuell tid.
Testfall **5**	**6,6** sekunder	**1,6** sekunder	Upptar **24,2%** av manuell tid.
Testprotokoll	**136,6** sekunder	**19,1** sekunder	Upptar **14,0%** av total manuell tid.

Ur tabell 13 kan man urskilja att det automatiskt utförda testprotokollet endast upptar 14% av den tid det tar för det manuellt utförda testprotokollet att genomföras. Detta är en avsevärd skillnad som skulle spara stora mängder tid för testarna på Askås I&R. Ytterligare utifrån tabell 13 så kan man se att de testfall som generellt tar längre tid att utföras resulteras i störst tidsbesparing. De testfall som resulterar i större besparing är då testfall 1 och testfall 2. Dessa testfall är sådana testfall som inkluderar ifyllning av fält, eller verifiering av ett flertal fält. Syftet med testfall 1 var att skapa en ny kundprofil hos butikssidan, vilket inkluderade ifyllnad av ett flertal fält för kundinformation. Testfall 2 hade därefter som syfte att verifiera att den information som blev inmatad av användaren i testfall 1, faktiskt arkiveras på administrationssidan. Det är förståeligt att dessa testfall får en avsevärd effektiviseringsgrad, då inmatning av fält är en tidskrävande process för testaren. Testfall 3, testfall 4 och testfall 5 i sin tur har en lägre effektiviseringsgrad då tidsskillnaden mellan dessa automatisk- och manuellt utförda testfall var mindre. Detta är också något som inte är förvånande, då automatisering resulterar i störst tidsbesparing hos mer tidskrävande testfall. Detta är intressant då det påverkar prioritering för automatisering av testfall hos företaget.

Det testprotokoll som Askås I&R utför vid var systemrelease i dagsläge består av 18 punkter, till skillnad från det testprotokoll som har blivit automatiserat i studien som endast bestod av fem punkter. Det är orealistiskt i dagsläget att automatisera alla 18 punkter, då många av dessa punkter kräver varierad input, samtidigt som några punkter är svåra att automatisera enligt systemarkitekten; se 4.2 *Intervjusvar från Systemarktitekten på Askås I&R*. Även om det är orealistiskt att automatisera hela testprotokollet i dagsläget så skulle en automatisering av exempelvis hälften av alla testfall resultera i en stor tidsbesparing, samtidigt som det skulle underlätta för testarna då färre manuella tester krävs.

De testfall som blev automatiserade i undersökningen valdes då de var relativt lätta att automatisera ur ett kodbaserat perspektiv. Dock, potentiellt så kan större delen av de 18 punkterna automatiseras under vidarearbete. En automatisering av alla 18 punkter kan kanske följa resultatet från testprotokollet som blev utfört under studien, och förhoppningsvis ta endast 14% (Tabell 13, sista raden) av den tid det tar för testprotokollet vid ett manuellt utförande.

6. Slutsatser

Syftet med uppsatsen var att undersöka samt implementera ett passande system för automatiserade funktionstestning till aktiebolaget Askås I&R. Utöver detta utfördes en undersökning för att framhäva just vilka testfall som bör prioriteras att automatiseras. De slutsatser som dras nedan baseras på den analys som gjorts av insamlad data, samt den uppmätta effektiviseringen av testprotokollen när det automatiskt utförda testprotokollet jämförs med det manuellt utförda testprotokollet.

6.1 U1 - Vilka testprotokoll är relevanta att automatisera för företaget?

De testprotokoll som bör prioriteras under automationen är de testprotokoll som resulterar i störst tidsbesparing för företaget. En automation av ett testprotokoll som innefattar testfall som utförs oftast vid ny systemrelease, samt realistiskt kan automatiseras, kommer att resultera i störst effektiviseringsgrad för företaget. Utifrån undersökningen bör alltså Askås I&R automatisera de testfall som upprepas oftast, samt de testfall som täcker stora/ centrala delar av systemet. Dessa testfall kommer potentiellt att hitta flest fel hos nya systemreleaser i framtiden.

6.2 U2 - Vilket programspråk för automatiska funktionstester är mest lämpligt för Askås I&R?

Utifrån den enkät som skickades till de anställda på Askås I&R så framgick det att javascript var det ledande programspråk som flest anställda besatt störst kunskaper inom. Det var senare detta programspråk som användes vid utveckling av det automatiserade testprotokollet som framgick i studien. Då Askås I&R är ett delvis webbaserat företag som bedriver webbutveckling så är det förståeligt att javascript var ett välkänt programspråk hos de anställda. Det är dock viktigt att tillägga att endast fem av åtta anställda deltog i enkätundersökningen. Vid ett större deltagarantal kunde utfallet från enkäten avvika från resultatet i som redovisas här.

6.3 U3 - Till vilken grad effektiviseras testningsrutinerna av automatisering av testfall?

Vid automatisering av det testprotokoll som användes under studiens gång så var effektiviseringsgraden stor. Det automatiskt utförda testprotokollet med fem testfall upptog endast 14% av den tid det tog att genomföra testprotokollet manuellt.

Studien har visat att full automation av specifika testfall kan resultera i stora tidsbesparingar för testaren. Vid full automation av alla 18 punkter så är det realistiskt att även det testprotokollet kommer att ha en liknande effektiviseringsgrad som testprotokollet i uppsatsen. Dock är full automatisering svårt i dagsläget, men realistiskt i framtiden. Även vid en automation av majoriteten av de 18 testfallen som utförs i dagens läge, så kommer tidsbesparingen att vara stor, speciellt med tanke på att testfallet utförs ofta vid varje systemrelease.

6.4 Självreflektion

Denna studie kan effektiviseras under ett antal punkter för framtida vidarestudier. Ett större deltagarantal vid enkätundersökningen skulle exempelvis precisera programspråksvalet. I denna studie så gjordes också endast en översikt över olika verktyg för automatiserade funktionstester som finns på marknaden. En djupare studie kunde potentiellt hitta ett mer effektivt verktyg som passar just Askås I&R.

Nästa steg för denna studie vore att effektivisera de testfall på 18 punkter som Askås I&R utför, så mycket som möjligt, för att få ett mer precist utfall av effektiviseringsgraden för de automatiserade testfallen. Alternativa större testfall kan också automatiseras för att få en större effektiviseringsgrad. De testfall som automatiserades i studien valdes då de var enklare att automatisera i studiens syfte. Automation av mer invecklade testfall bör realistiskt ha en större effektiviseringsgrad jämfört med de som blev automatiserade i uppsatsen.

Referenser

Tryckta källor (inkl. pdf)

Ammann, P. & Offutt, J. (2008). *Introduction to Software testing*. Cambridge University Press.

García, B. & Dueñas, J. C. (2014). *Automated Functional testing based on the Navigation of Web Applications*. Cornell University Library.
Tillgänglig: https://arxiv.org/abs/1108.2357.

Gojare, Satish., Joshi, Rahul. & Gaigaware, Dhanashree. (2015). Analysis and Design of Selenium WebDriver Automation testing Framework*. Procedia Computer Science*, vol. 50, sid 341-346. Tillgänglig:
http://www.sciencedirect.com/science/article/pii/S1877050915005396?via%3Dihub

Gundecha, U. (2015). *Selenium testing Tools Cookbook.* Second edition. Packt Publishing.

Holmes, A. & Kellogg, M. (2006). *Automating Functional Tests Using Selenium*. Proceedings of AGILE 2006 Conference (AGILE'06). IEEE. Tillgänglig:
https://pdfs.semanticscholar.org/9c9c/2cdabc83bdb503766bf06941513194ac5419.pdf

Myers, G J., Badgett, T. & Sandler, C. (2012). *The Art of Software testing*. Third edition. Hoboken, New Jersey: John Wiley & Sons.

Patel, R. & Davidson, B. (2014). *Forskningsmetodikens grunder.* Fjärde upplagan. Lund: Studentlitteratur.

Rothermel, G., Untch, R H,. Chu, C. & Harrold, M J. (1999). Test Case Prioritization: An Empirical Study*. Proceedings of the International Conference on Software Maintenance, Oxford, UK, September 1999*. Tillgänglig:
https://pdfs.semanticscholar.org/131a/32c965c16880f273d00fec7a61e33884544f.pdf

Rubin, J. & Chisnell, D. (2008). *Handbook of usability testing: how to plan, design and conduct effective tests*. (2. ed.) Indianapolis, Indiana: Wiley Pub.

Ryber, Torbjörn. (2007). *Essential Software Test Design*. Dublin: Uniquepublishing.ie.

Tarlinder, A. (2017). *Developer Testing. Building Quality into Software*. Crawfordville, Indiana: Pearson Education.

Webbsidor

Askås I&R. (2017). *Aktiebolaget Askås I&R hemsida*. [Elektronisk].
Tillgänglig: http://www.Askås I&R.se/sv/om-Askås I&R.html [2017-03-08]

Facebook. (2017). *Facebook Newsroom Site.* [Elektronisk].
Tillgänglig: http://newsroom.fb.com/company-info/ [2017-03-07]

ITD. (2017). *Populära CMS*. [Elektronisk].
Tillgänglig: http://www.itd.li/j15/index.php/component/content/article/6-vad-aer-cms [2017-03-28]

Selenium HQ. (2017). *Selenium Projects*. [Elektronisk].
Tillgänglig: http://www.seleniumhq.org/projects/ [2017-03-07]

Selenium HQ JS. (2017). *Javascript (Node) kodexempel.* [Elektronisk].
Tillgänglig: https://github.com/SeleniumHQ/selenium/tree/master/javascript/node/selenium-webdriver [2017-03-27]

Software testing Studio. (2017). *What Is Meant By Selenium WebDriver?* [Elektronisk]
Tillgänglig: http://www.softwaretestingstudio.com/selenium-webdriver/ [2017-03-15]

WebdriverIO. (2017). Webbplats för inlärning av NodeJS. [Elektronisk].
Tillgänglig: http://webdriver.io/. [2017-05-25]

Referenser för undersökning av potentiella verktyg

Csjobb.idg.se (2017). *Testautomatiserare på Sogeti*. [Elektronisk]. Tillgänglig:
https://csjobb.idg.se/jobb-info/57441727/testautomatiserare-med-passion-for-kvalite/ [2017-05-05]

Softwareqatest.com (2017). *Web Site Test Tools and Site Management Tools.* [Elektronisk].
Tillgänglig: http://www.softwareqatest.com/qatweb1.html [2017-01].

Testingtools.com (2017). *Automated Web testing Tools.* [Elektronisk].
Tillgänglig: http://www.testingtools.com/test-automation/ [2017-01]

Tieto.se (2017). *Test automation - functional testing solution*. [Elektronisk]. Tillgänglig:
https://www.tieto.com/services/application-development/testing-services/test-automation
[2017-03-15]

Bilaga – Enkätens utformning.

FRÅGOR SVAR 4

Avsnitt 1 av 2

Enkät - Testning på Askås

Syftet med denna enkät är att samla in information om testningsrutiner på Askås. Enkäten innehåller frågor om erfarenheter, ditt testningsarbete och dina åsikter om testningsrutiner.

Tack för att du deltar i undersökningen angående testning på Askås I&R! Vi finner ditt medverkande som mycket hjälpsamt och hoppas att du tar dig tid att svara på alla frågor! Ta den tid du känner på dig och svara ärligt.

Vill du som deltagare förbli anonym för att förhindra att svaren du skriver kan spåras tillbaka till dig, kommer du endast att bli refererad till som exempelvis 'Anställd 1'.

Observera: Denna intervju innehåller ej känsliga frågor, men valet att vara anonym finns!

Namn (lämna tomt om du vill vara anonym)

Kort svarstext

Vad är din nuvarande position på Askås I&R bortsett från att utföra tester? Vill du förbli anonym och känner att din position kan vara känslig information så kan du ignorera frågan.

Lång svarstext

[T1] Hur mycket tid (generellt) lägger just du på test av system i veckan / ny systemrelease?

Lång svarstext

[T2] Tror du att Askås I&R borde utveckla sina testningsrutiner?

Lång svarstext

[T3] Hur tror du att testningsrutinerna kan effektiviseras?

Lång svarstext

[T4] Borde Askås I&R spendera mer eller mindre resurser (tid) på att testa sin produkt?

Lång svarstext

FRÅGOR SVAR 4

Avsnitt 2 av 2

Programspråk och erfarenheter

Beskrivning (valfritt)

[P1] Har du tidigare programspråk-erfarenheter bortsett från Perl? Bocka i den erfarenhet du har kring respektive språk.

Rad 1.	Java	Kolumn 1.	Ingen Erfarenhet
Rad 2.	C#	Kolumn 2.	Lägt Erfaren
Rad 3.	JavaScript	Kolumn 3.	Erfaren
Rad 4.	JavaScript (Node)	Kolumn 4.	Mycket Erfaren
Rad 5.	Python		
Rad 6.	Ruby		

[P2]Har du erfarenhet av andra programspråk, i så fall vilket / vilka?

Lång svarstext

[P3] Känner du att språket Perl är ett vitalt språk för dig när du utför testprotokoll?

Lång svarstext

[P4] Skulle du kunna tänka dig att lära ett nytt programspråk om det innebar mer effektiv testning?

Lång svarstext