EmoTouch

Nästa generations analysverktyg för mobilapplikationer

LUNDS UNIVERSITET
Lunds Tekniska Högskola

LTH Ingenjörshögskolan vid Campus Helsingborg
Datateknik

Examensarbete:
Pontus Hagberg
Petter Lundahl

LTH Ingenjörshögskolan vid Campus Helsingborg
Lunds universitet
Box 882
251 08 Helsingborg

LTH School of Engineering
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

Tryckt i Sverige
Media-Tryck
Biblioteksdirektionen
Lunds universitet
Lund 2014

Sammanfattning

Det här arbetet utforskar möjligheter att konstruera ett analysverktyg som kartlägger användarbeteenden i mobilapplikationer, med avsikt att detektera frustrerat beteende. Projektet har genomförts i samarbete med företaget Tactel AB, som ser en kommersiell potential i ett analysverktyg som kan ge applikationsutgivare information om användares frustrationsnivåer i realtid. Projektets målsättning var att utveckla en prototyp av ett sådant analysverktyg, som mäter relevanta användarbeteenden, detekterar avvikelser och visualiserar information om avvikande beteenden hos stora grupper användare.

Under projektet utvecklades en prototyp, *EmoTouch*, bestående av två delar: Android-modulen *EmoTouch Android SDK* och webbapplikationen *EmoTouch Dashboard*. Modulen mäter olika parametrar i användarens interaktion med en applikation. Parametrar har valts ut för mätning baserat på resultat av befintlig forskning, där samband mellan olika användarbeteenden och användares emotionella reaktioner har undersökts. Vid mätning byggs en individuell beteendeprofil upp, vilken används för att detektera parametervärden som avviker från användarens normala beteende. Genom analys av sådana avvikelser kategoriseras beteendet som frustrerat eller icke-frustrerat.

För att kunna sammanställa data från en stor grupp användare har *EmoTouch* konstruerats som ett tillägg till Google Analytics. *EmoTouch Dashboard* hämtar och visualiserar denna data. För att undersöka *EmoTouchs* förmåga att identifiera frustration genomfördes tester med försökspersoner. Försökspersonerna fick interagera med en Android-applikation, först på ett normalt sätt och därefter med ett simulerat frustrerat beteende.

Resultatet av testerna visade ett samband mellan andelen avvikande parametervärden och ett simulerat frustrerat beteende, vilket indikerar att det finns möjlighet att detektera frustration utifrån *EmoTouchs* metod för att kartlägga användarbeteenden. *EmoTouch* kunde inte särskilja mellan normalt och simulerat frustrerat beteende, men analys av testresultatet har visat hur metoden för att identifiera frustration skulle kunna förbättras.

Projektet har visat hur användarbeteenden kan kartläggas i en godtycklig Android-applikation, hur avvikande beteenden kan detekteras och hur det utifrån dessa avvikelser går att visualisera information om stora användargruppers beteenden. Projektet har även visat hur Google Analytics kan utnyttjas som plattform för att konstruera ett mer kraftfullt verktyg.

Nyckelord: Android, Användarbeteende, Google Analytics, Frustration, Analysverktyg för mobilapplikationer.

Abstract

This thesis explores the possibility to develop a tool for mobile analytics, which maps user behavior with the intent to detect frustration. The project was carried out in cooperation with the company Tactel AB. They see a great commercial potential in a tool that can provide application publishers with information about users' frustration levels in real-time. The objective of the project was to develop a prototype for such a tool, which measures relevant user behavior, detects deviations and visualizes information about behavior deviations in large user populations.

During the project a prototype was developed, *EmoTouch*, consisting of two parts: the Android module *EmoTouch Android SDK* and the web application *EmoTouch Dashboard*. The module measures various parameters in the user's interaction with an application. Parameters were selected for measurement based on the results of previous studies, where correlations between user behavior and affective responses were examined. A user-specific profile is established as behavior is measured. The profile is used to detect parameter values that deviate from the user's normal behavior. Through analysis of such deviations, behavior is categorized as either frustrated or non-frustrated.

In order to enable *EmoTouch* to combine data from large user populations, it was implemented as a plugin to Google Analytics. *EmoTouch Dashboard* retrieves and visualizes this data. In order to examine *EmoTouch's* ability to detect frustration, tests were conducted with test subjects. The test subjects interacted with an Android application, first in a normal way, and then with a simulated frustrated behavior.

The results of the tests showed a relation between the proportion of deviating parameter values and simulated frustrated behavior, which indicates the possibility of detecting frustration based on *EmoTouch's* method for mapping user behavior. *EmoTouch* could not differentiate between normal and simulated frustrated behavior, however analysis of the test results revealed ways of improving the method for identifying frustration.

The project has shown how user behavior can be mapped in an arbitrary Android application, how deviating behavior can be detected and how information about the behavior of large user populations can be visualized. The project has also shown how Google Analytics can be utilized as a platform in order to create a more powerful tool.

Keywords: Android, User behavior, Google Analytics, Frustration, Mobile analytics

Förord

Detta arbete har genomförts som ett examensarbete på högskoleingenjörsutbildningen i datateknik vid Lunds Tekniska Högskola. Utgångspunkten för arbetet var en vision hos Tactel AB och arbetet har utförts i samarbete med företaget.

Vi skulle vilja tacka vår handledare Karl Slättorp för hans hjälp och kommentarer, samt Olof Råborg och alla andra på Tactel AB som har bidragit till att göra detta arbete möjligt. Vi vill också tacka vår examinator Lise Jensen för alla värdefulla synpunkter under arbetets gång, och Elin Anna Topp som bistod oss med sin kompetens inom maskinlärande.

Malmö 2014-05-23

Pontus Hagberg
Petter Lundahl

Innehållsförteckning

1 Inledning

1.1 Bakgrund

Det finns ett stort intresse bland applikationsutgivare att veta vilka deras användare är och hur dessa interagerar med deras produkt. Sådan information har ett stort värde både för marknadsföringssyften och för att bättre anpassa produkten för användarna. Det finns en rad verktyg på marknaden idag (Appsee, Google mobile analytics, opentracker osv) som ger utgivaren en stor mängd information om vilka användarna är och vilka problem användarna upplever vid användning. En möjlig vidareutveckling av denna typ av analysverktyg är att lägga till ytterligare en dimension: Möjligheten att mäta användarens emotionella reaktioner vid användningstillfället. De tänkbara användningsområdena för ett sådant verktyg är många.

Företaget Tactel AB har en vision om att utgivare av applikationer ska kunna använda ett sådant verktyg för att upptäcka eventuell frustration hos stora grupper användare i realtid. De ser en stor kommersiell potential i att utveckla en sådan produkt för marknaden. Genom att övervaka frustrationsnivån hos olika användargrupper går det att skapa en större förståelse för användarnas beteende och vidta lämpliga åtgärder för att minska frustrationen.

Men hur kan ett sådant verktyg konstrueras? Det finns flera studier som har lyckats påvisa ett samband mellan en användares beteende och dennes känslotillstånd vid användandet av ett datorprogram eller en mobilapplikation. Om det med hjälp av ett verktyg går att upptäcka beteenden som avviker ifrån användarens normala beteenden, kan detta ge en indikation om användarens emotionella reaktioner.

1.2 Syfte och målsättning

Syftet med projektet var att undersöka möjligheten att skapa ett system som i realtid kan upptäcka avvikande användarbeteenden som kan tolkas som tecken på frustration.

Målsättningen var att utveckla en prototyp av ett analysverktyg som kartlägger användarbeteeenden i mobilapplikationer för att i realtid kunna upptäcka ett simulerat frustrerat beteende hos applikationsanvändare. Ett delmål var att undersöka vilken typ av användarbeteenden som var tekniskt möjliga att registrera och vilka som var relevanta för att dra slutsatser om en användarens emotionella reaktioner. För att uppnå detta behövde det även undersökas hur användarbeteenden kan registreras, rapporteras, kartläggas och analyseras för att upptäcka beteendeavvikelser i en befintlig mobilapplikation.

Den långsiktiga visionen med projektet var att de resultat som uppnåddes ska kunna användas som utgångspunkt vid konstruktionen av ett analysverktyg för marknaden, som har potentialen att markera begynnelsen av en helt ny generation av analysverktyg.

Prototypen som utvecklades har arbetsnamnet *EmoTouch* och benämns så i följande delar av rapporten. *EmoTouch* består av en Android-modul som benämns *EmoTouch Android SDK* och en webbapplikation som benämns *EmoTouch Dashboard*.

1.3 Problemformuleringar

- Hur kan användarbeteenden kartläggas vid användning av en mobilapplikation?
- Hur kan dessa kartlagda beteenden användas för att detektera avvikande beteenden i realtid?
- Hur kan dessa beteendeavvikelser användas för att identifiera ett simulerat frustrerat beteende?
- Hur kan information om avvikande beteende hos stora grupper användare visualiseras i realtid?

1.4 Avgränsningar

Projektets huvudsakliga fokus var att utveckla en fungerande prototyp som i realtid detekterar och registrerar avvikelser från en enskild användares normalbeteende. Användarbeteenden som kan vara möjliga indikationer på frustration valdes utifrån resultaten av befintliga studier inom områdena kognitionsvetenskap och människa-datorinteraktion. Dessa studier analyserades inte ytterligare då det inte låg inom omfånget för arbetet.

Prototypen utvecklades för Android. Detta på grund av att Android är ett väldokumenterat operativsystem för mobila enheter och därför är lämpat som utvecklingsplattform.

Frustration är inte ett helt entydigt begrepp men ordet används i det här arbetet för att beteckna negativa emotionella reaktioner. Begreppet omfattar exempelvis vrede, irritation och stress. Eftersom visionen med projektet var att detektera frustration kommer det att förutsättas att en användare kan vara i ett av två olika känslotillstånd: Ett frustrerat tillstånd och ett icke-frustrerat tillstånd.

2 Definitioner

Term	Definition
Användarbeteende	Användarens fysiska interaktion med sin telefon eller surfplatta. Hit räknas inte komplexa beteendemönster som till exempel användarens språkbruk eller i vilka sekvenser användaren brukar använda olika funktioner.
Normalbeteende	Användarens genomsnittliga beteende uppmätt under en längre tidsperiod.
Avvikande beteende	Beteende som mätbart avviker från det definierade normalbeteendet.
Simulerat frustrerat beteende	Det användarbeteende som en försöksperson uppvisar när denne uppmanas att bete sig som i ett frustrerat tillstånd. Detta tillstånd används vid test av EmoTouch förmåga på grund av svårigheter att provocera fram en äkta känsla av frustration hos en försöksperson.

3 Metod

Projektet delades in i två faser, Fas 1 och Fas 2. Fas 1 ägnades åt undersökningar med syfte att samla in den information som var nödvändig för att fastställa en målbild inför Fas 2, då arbetet med att utveckla en prototyp tog vid.

Fas 1 utgjordes av följande moment:

- En kartläggning av liknande analysverktyg på marknaden.
- En undersökning av Android-plattformen och dess tekniska förutsättningar i relation till projektet. Detta inkluderade en förstudie där en enkel applikation för tester utvecklades.
- Läsning av relevanta forskningsartiklar som undersöker eventuella samband mellan mätbara användarbeteenden och användarens känslotillstånd.

Fas 1 avslutades med att målbilden utformades utifrån den information som samlats in. Målbilden låg till grund för arbetet som utfördes i Fas 2.

Fas 2 utgjordes av ett utvecklingsprojekt enligt en iterativ projektmodell i nära samarbete med värdföretaget. Under utvecklingsprojektet implementerades *EmoTouch* enligt målbilden som togs fram under Fas 1. Först implementerades Android-modulen *EmoTouch Android SDK* och sedan webbapplikationen *EmoTouch Dashboard*. Fasen avslutades med testning med försökspersoner, då *EmoTouchs* förmåga att kartlägga användarbeteenden och identifiera simulerat frustrerat beteende undersöktes.

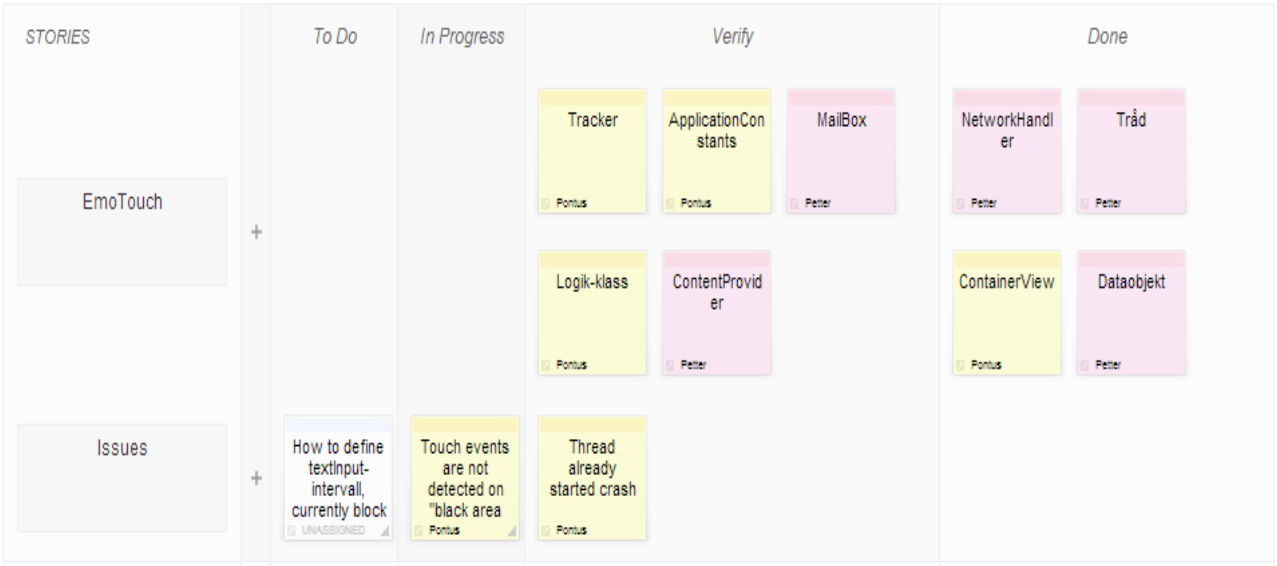
3.1 Projektmodell

Projektmodellen utformades utifrån projektets förutsättningar vilka är följande:

- Ett mycket litet utvecklingsteam omfattande två personer som har möjlighet att arbeta på samma plats under hela projektet. Detta medförde att det inte fanns ett stort behov av omfattande hjälpmedel för att skapa transparens och visualisera arbetsflödet.
- Det fanns en vision hos beställaren om vilka mål produkten skulle realisera, men få specifika krav på konkret funktionalitet. Detta innebar att det fanns ett behov att validera resultatet av arbetet kontinuerligt.
- Ett experimentellt användningsområde. Det var nödvändigt att utföra någon form av test med försökspersoner för att överhuvudtaget veta om det färdiga systemet uppfyller beställarens förväntningar.

Projektmodellen definierades på följande sätt:

- Arbetsflödet visualiserades med hjälp av en flödestabell inspirerad av ett Scrumboard där olika arbetsuppgifter flyttades mellan kolumnerna “Att göra”, “Pågående”, “Verifiera” och “Färdigt”. Arbetsuppgifter kunde vara implementering av ny funktionalitet eller hantering av buggar. Detta var den enda dokumentationen av projektet utöver denna rapport. Figur 1 visar flödestabellen vid en tidpunkt i projektet.
- Iterationerna hade ingen exakt definierad längd, men omfattade ungefär sex veckor. Efter en genomförd iteration demonstrerades en fungerande prototyp för beställaren, och nästa iteration planerades efter diskussion med beställaren.



Figur 1: Projektets flödestabell under utvecklingsprocessen

3.2 Rapportens struktur

I den här rapporten beskrivs projektets två faser följt av ett sammanfattande avsnitt.

I redogörelsen för Fas 1 (kapitel 4) beskrivs resultaten av undersökningarna och utifrån dessa motiveras de beslut som tagits inför Fas 2. I målbilden (kapitel 5) beskrivs utgångspunkten för det system som konstruerades under projektets andra fas.

I redogörelsen för Fas 2 (kapitel 6) beskrivs det utvecklade systemets arkitektur och utvecklingsprocessen. I testkapitlet (kapitel 7) beskrivs resultat och slutsatser av tester som har utförts. Därefter följer en diskussion om etiska överväganden kring problemområdet (kapitel 8). I den sammanfattande delen (kapitel 9) diskuteras resultatet av arbetet och i vilken utsträckning problemformuleringen har besvarats. Den här delen innehåller också en diskussion om möjligheter till vidareutveckling av systemet.

4 Fas 1

4.1 Liknande tjänster

Det finns i dagsläget flera analysverktyg för mobilapplikationer som kan samla in information om dess användare, för att sedan presentera informationen för applikationens utgivare. Det som följer är en beskrivning av tre sådana analysverktyg för mobila enheter. Dessa har valts ut för att de har ett stort fokus på användarens beteende och interaktion med applikationen. Det finns många liknande verktyg som fokuserar mer på att samla in data om vilka användarna är, eller på att analysera utifrån konkreta affärsmål, exempelvis hur många procent av användarna som genomför ett köp. Denna typ av verktyg är dock av mindre intresse i förhållande till arbetets inriktning.

De tre verktygen som beskrivs i detta kapitel är Google Analytics, Appsee och Countly. Dessa tre verktyg tillhör alla gruppen av verktyg som samlar in data om användarnas beteende. Google Analytics är relevant eftersom det är ett etablerat och välanvänt analysverktyg. Appsee visar möjligheten för verktyg som analyserar användarupplevelsen i mobilapplikationer genom att automatiskt samla data om användarens beteende. Eftersom Countly är ett open-source projekt ger det en bild av hur koden för ett sådant verktyg kan vara uppbyggd, det synliggör till exempel verktygens uppdelning i server- och moduldel.

4.1.1 Google Analytics

Google Analytics är ursprungligen ett verktyg för att administratörer av webbsidor ska få tillgång till statistik om sidans besökare. Det finns även ett SDK för Android och ett för iOS-applikationer tillgängligt för utvecklare som vill samla information om sina applikationers användare. Verktyget kan till exempel användas för att ta fram information om hur många aktiva användare en applikation har, var i världen dessa befinner sig, vilka olika funktioner i applikationen som används och hur ofta applikationen kraschar. Verktyget erbjuder möjligheten att skapa egendefinierade händelser som är av speciellt intresse och som ska övervakas och loggas, exempelvis att användaren trycker på en viss knapp i applikationens grafiska gränssnitt (Google Inc. 2013).

Det går även att ta fram mer detaljerad information om hur användare interagerar med applikationen, exempelvis vilka delar av applikationen som besökts och i vilken ordning detta skett (Google Inc. 2014a).

Den insamlade informationen är tillgänglig via Google Analytics webbapplikation och visualiseras genom grafer och tabeller.

4.1.2 Appsee

Appsee är ett verktyg som ger utvecklare möjligheten att följa sina användares beteenden, både på individuell nivå och på en mer övergripande nivå. Verktyget finns i dagsläget endast till iPhone men en version till Android är planerad att släppas. Appsee består av två delar, ett SDK som integreras i en iPhone-applikation och en webbapplikation som samlar in och presenterar informationen om användarna (Appsee 2014a).

Genom att integrera Appsee i en applikation samlas en stor mängd information om användaren automatiskt in och laddas upp till en server. Verktyget samlar både in information om vem användaren är och vad denne gör. Användarens handlingar i applikationen spelas in med video och laddas upp så att utvecklaren kan se exakt hur individuella användare interagerar med applikationen. Verktyget loggar dessutom automatiskt tidpunkt och typ av händelser i applikationen, till exempel gester, knapptryck, sidbyten och krascher. Det finns även en funktion som skapar så kallade Touch Heatmaps för varje skärm i applikationen, som visar var och hur frekvent användare har tryckt på skärmen (Appsee 2014b).

Via webbapplikationen går det att i realtid följa individuella användare och att se statistik över hur applikationen används på en större skala.

4.1.3 Countly

Countly är ett open-source verktyg för analys av hur mobilapplikationer används. Verktyget består av ett SDK som integreras i en applikation och en webbapplikation. Countly SDK finns till Android, Blackberry, Windows Phone och iPhone. Genom att integrera Countly i en applikation samlas en mängd information om användaren in, till exempel mobiloperatör, land, operativsystem och antal användningssessioner. Utöver detta finns det möjligheter för utvecklaren att undersöka hur applikationen används genom egendefinierade händelser. Genom att lägga till en egendefinierad händelse vid ett visst skede i applikationen, till exempel när användaren trycker på en viss knapp, så loggas detta och blir en del av användningsstatistiken (Countly 2014).

All information som samlas in laddas upp till en server och presenteras via en webbapplikation där det går att se en mängd statistik över en applikations användare. Till exempel visas statistik om hur ofta ens egendefinierade händelser inträffar (Countly 2014).

4.1.4 Sammanfattning

De tre verktyg som beskrivits representerar den typ av analysverktyg som *EmoTouch* har som utgångspunkt. De är analysverktyg som låter applikationsutgivare se information om hur användare interagerar med deras produkter, men inget av verktygen försöker tolka användarens emotionella reaktioner. Det gemensamma för dessa verktyg är att de bygger på en struktur där en modul integreras i applikationens källkod och sedan kommunicerar med en server via internet. Servern lagrar information om applikationens olika användare som görs tillgänglig för applikationens utgivare via en webbapplikation. Eftersom Countly är open-source kan dess konstruktion studeras och användas som underlag vid implementeringen av *EmoTouch*. Både Google Analytics och Countly har moduler tillgängliga för Android-applikationer. *EmoTouch Android SDK* kan utformas på ett sätt så att integreringen av modulen i en befintlig applikations källkod görs på samma sätt som Countly och Google Analytics.

Samtliga tre verktyg erbjuder möjligheten för utvecklaren att lägga in någon form av egendefinierade händelser på valfria ställen i källkoden, så att en viss händelse i applikationen orsakar ett anrop till servern. Om denna funktionalitet implementeras i *EmoTouch* kan det underlätta analysen av data om avvikande beteende genom att sätta datan i ett sammanhang. Det kan vara relevant för utgivaren av applikationen att veta vilka moment i applikationen som orsakar mest frustration hos användare.

4.2 Relaterad forskning

Det finns flera exempel på studier som har syftat till att undersöka sambanden mellan en användares interaktion med ett datorprogram och användarens känslotillstånd. I det följande avsnittet sammanfattas resultaten av fem sådana studier. Dessa har valts ut för att de presenterar konkreta resultat som är tillämpningsbara vid utvecklandet av ett system som kan detektera indikationer på frustration hos användare av mobilapplikationer.

De undersökta studierna har pekat ut ett antal mätbara parametrar som kan vara användbara för att göra förutsägelser om en användares känslotillstånd.

Dessa variabler är:

- Tryckarea - Stor area vid skärmtryck kan indikera frustration. (Gao, Bianchi-Berthouze & Meng 2012)
- Varaktighet av gester och skärmtryck - Kort varaktighet kan indikera frustration (Gao, Bianchi-Berthouze & Meng 2012) och stress. (Anthony, Carrington, Chu, Kidd, Lai & Sears 2011)
- Tryckhårdhet - Större hårdhet vid tryck på skärm vid upplevd frustration. (Mentis 2002)
- Längd av gester - Kortare längd av gester kan indikera stress. (Anthony et al. 2011)
- Hastighet av gester (längd i pixlar dividerat med varaktighet) - Gester med högre hastighet kan indikera stress. (Anthony et al. 2011)
- Skrivhastighet - Låg skrivhastighet kan indikera ilska. (Tsihrintzis, Virvou, Alepis & Stathopoulou 2008)
- Raderingsfrekvens - Hög frekvens av radering av text kan indikera ilska. (Tsihrintzis et al. 2008)
- Antal musklick - Fler musklick kan indikera frustration. (Scheirer, Fernandez, Klein & Picard 2001)

För att detektera ett frustrerat beteende måste det definieras vad som utgör ett normalt beteende. För att definiera ett normalt beteende måste en av två modeller väljas: Ett användarspecifikt eller ett användaroberoende normalbeteende. Den användarspecifika modellen innebär att varje enskild användare har en egen normalprofil som formas efter dennes vanliga beteenden. Den användaroberoende modellen innebär att det bara finns en normalprofil som formas efter en stor grupp användares genomsnittliga beteenden. De studier som har jämfört resultaten av de två olika modellerna konkluderar att den användarspecifika modellen uppnår en högre grad av korrekthet än den användaroberoende modellen. (Mentis 2002; Gao, Bianchi-Berthouze & Meng 2012)

Här följer en mer detaljerad redogörelse för de olika studierna och deras resultat.

4.2.1 Skärmtryck och gester

Ett av experimenten beskrivs av Gao, Bianchi-Berthouze och Meng (2012) och det utfördes genom att låta en grupp försökspersoner spela ett mobilspel (en variant av spelet Fruit Ninja) för iPod Touch, samtidigt som information om deras fingergester sparades av applikationen. De fick sedan besvara frågor om sitt känslotillstånd. Syftet med studien var att undersöka möjligheten att utifrån en användares fingergester automatiskt detektera ett av följande fyra möjliga känslotillstånd: Frustration, Uttråkning, Upphetsning och Avslappning. Försökspersonernas känslotillstånd definierades genom självuppskattning.

Detta visade sig möjligt med en korrekthet på 69-77%. Av de fyra känslorna som undersöktes visade sig just frustration vara den som kunde identifieras med högst grad av korrekthet. Under experimentet användes totalt 16 olika mätparametrar, exempelvis varaktigheten, längden (mätt i pixlar) eller tryckytan av en fingergest. De parametervärden som visade sig vara de tydligaste indikationerna på frustration var:

- Gester och skärmtryck med stor tryckarea
- Gester och skärmtryck med kort varaktighet

De förklarar att det egentligen är tryckhårdhet (“touch pressure” på engelska) de är intresserade av när de mäter tryckarea. Men eftersom pekskärmen inte tillhandahåller denna information använder de tryckarea som en indikation på hur hårt användaren trycker. I det här arbetet kommer det också att antas att det finns ett starkt samband mellan tryckarea och tryckhårdhet.

För att detektera att en parameter antog ett värde som avvek från det normala testades två olika modeller för att identifiera avvikande värden: den ena jämförde med användarspecifika mätdata, den andra jämförde med mätdata som var sammanställda utifrån samtliga försökspersoners genomsnittliga användarbeteende. Med den användarspecifika modellen kunde känslotillstånd identifieras med en något högre grad av korrekthet i jämförelse med den användaroberoende. Resultaten skiljde sig dock endast med ett fåtal procentenheter.

Ett annat experiment inom området beskrivs av Mentis (2002) och utfördes genom att låta en grupp försökspersoner utföra olika arbetsuppgifter i Microsoft Word på en dator utrustad med en pekplatta. Samtidigt mättes och sparades data om tryckhårdheten i deras fingertryck på pekplattan och efteråt intervjuades de och fick svara på frågor om huruvida de upplevt frustration

under utförandet av de olika arbetsuppgifterna. Syftet var att hitta ett samband mellan upplevd frustration och tryckhårdhet.

Resultatet visade att försökspersonerna tryckte hårdare på pekplattan vid tillfällen då de upplevde frustration. Det visade sig också att den genomsnittliga tryckhårdheten och den genomsnittliga avvikelsen från denna i frustrerade situationer skiljer sig stort mellan de olika försökspersonerna. Utifrån denna observation argumenterar artikeln för att detektion av frustration måste göras genom jämförelse med användarspecifika mätdata, och inte genom jämförelse med den genomsnittlige användarens mätdata.

Anthony et al. (2011) har undersökt eventuella samband mellan försökspersoners upplevda stress och karakteristiken av deras gester då de interagerar med ett program på en surfplatta med hjälp av en pekpenna. Programmet innehöll olika stressande moment och försökspersonernas stressnivå uppmättes med instrument som mäter fysiologiska stressymptom såsom förhöjd hudtemperatur och puls. Under tiden sparade programmet data om karakteristiken av deras gester. Totalt 10 olika parametrar analyserades, bland annat gesters varaktighet, tryckhårdhet och hastighet (längd i pixlar dividerat med varaktighet).

Resultatet visar att ett stressat tillstånd kan indikeras av följande parametervärden:

- En signifikant minskning i varaktigheten av gester
- En signifikant minskning i längden av gester
- En marginellt signifikant ökning i hastigheten av gester

I en artikel av Scheirer et al. (2001) beskriver författarna hur de medvetet skapat en frustrerande situation för att undersöka fysiologiska och beteendemässiga reaktioner. 24 stycken försökspersoner fick under en timmes tid spela ett spel på en dator där till synes slumpmässiga fördröjningar lagts in vid vissa musklick. Fördröjningarna hade utformats på ett sätt så att det uppfattades som att musen slutade fungera. Tillsammans med det faktum att spelet gick på tid bidrog dessa fördröjningar till att skapa en frustrerande situation. Under experimentet mättes testpersonernas interaktion med musen i form av antal musklick. Genom att jämföra antalet musklick vid de frustrerande situationerna i förhållande till den normala spelsituationen fann författarna en viss ökning av klick vid fördröjningar. 17 av 24 testpersoner gjorde flera extra musklick vid fördröjningarna.

4.2.2 Inmatning av text

I en studie av Tsihrintzis et al.(2008) undersökte författarna om information om hur en användares interaktion med en dators tangentbord kan ge ledtrådar om dennes känslotillstånd. Målet med studien var att undersöka möjligheter att använda data från tangentbordsanvändning som ett komplement till känsloidentifikation genom bildanalys. Studien genomfördes med en empirisk undersökning med frågeformulär där 100 personer med olika bakgrund fick svara på frågor om deras känslotillstånd vid datoranvändning. De tillfrågade fick uppskatta hur deras tangentbordsbeteende påverkades av olika känslotillstånd.

De beteenden som studien undersökte var skrivhastighet, raderingsfrekvens, tryck på orelaterade tangenter och brist på användning av tangentbordet. Resultatet av studien pekade på att glädje, ilska och sorg hade en koppling till dessa beteenden. Det känslotillstånd som oftast gick att identifiera (74% korrekthet) var ilska. De beteenden som kopplades till ilska var när användaren: skriver långsammare än vanligt (27% av de tillfrågade), raderar text oftare än vanligt (60% av de tillfrågade) , trycker på orelaterade tangenter (40% av de tillfrågade) och inte använder tangentbordet alls (27% av de tillfrågade).

4.2.3 Sammanfattning

I de fem studierna beskrivs olika mätbara användarbeteenden som har använts för att detektera känslotillstånd med varierande resultat. Gao, Bianchi-Berthouze och Meng (2012) presenterade i sin studie en hög grad av korrekthet vid detektion av frustration hos en användare. Experimentet utfördes dock genom interaktion med ett mobilspel och det är värt att ifrågasätta om användare agerar ut sin frustration på samma sätt vid interaktion med en mindre känslomässigt engagerande typ av applikation, exempelvis en bankapplikation. Spelet bygger på att användaren utför långa fingergester över hela enhetens skärm, och det är utifrån informationen om dessa gester som indikationer på känslotillstånd har detekterats. Resultatet borde därför inte vara helt applicerbart på en applikation vars gränssnitt inte baseras på att användaren utför långa fingergester.

Både Tsihrintzis et al.(2008) och Scheirer et al. (2001) utförde experiment med persondatorer och alltså inte med mobila enheter eller andra enheter utrustade med pekskärm. Detta innebär att det är osäkert hur mycket studierna säger om frustrerade mobilanvändares typiska beteenden.

Beteendet som Tsihrintzis et al.(2008) benämner som tryck på orelaterade tangenter ges ingen entydig definition. Eftersom innebörden av “orelaterade

tangenter” kan tolkas på många olika sätt så anses inte denna parameter som relevant för arbetet.

4.2.3.1 Modell för normalprofil

Mentis (2002) argumenterade för den användarspecifika modellen vid detektion av frustration, vilket grundades på att olika användares normalbeteenden skiljde sig stort. Gao, Bianchi-Berthouze och Meng (2012) presenterade marginellt bättre resultat för detektion av känslotillstånd med en användarspecifik modell jämfört med en användaroberoende. I detta arbete antas att olika människor kan interagera med en mobilapplikation på helt olika sätt beroende på ålder, teknikvana och personliga egenskaper som temperament, tålamod och koncentrationsförmåga.

Ett argument för att använda en användaroberoende modell är att normalprofilen kommer att utformas fortare och baseras på större datamängder. Vid användning av en användarspecifik modell kommer det antagligen behövas ett visst antal användningstillfällen för att utforma normalprofilen för en ny användare, innan det går att detektera betydelsefulla avvikelser. Vid användning av en användaroberoende modell kan avvikelser detekteras redan vid en användares första användningstillfälle, förutsatt att tillräckligt många andra användare har använt applikationen tidigare.

EmoTouch utnyttjar en användarspecifik modell. Detta motiverades med att det bör innebära en högre grad av korrekthet vid identifiering av beteenden som indikerar frustration.

4.3 Teknisk plattform: Android

Android är ett open-source operativsystem baserat på Linux som idag är världens ledande operativsystem för mobila enheter. Applikationer till Android utvecklas i Java med hjälp av Androids utvecklingsverktyg ADT (Android Development Tools) (Google Inc. u.å.a).

Applikationer i Android består av ett par grundläggande komponenter som har olika syften. De huvudsakliga komponenterna är Activity och Service, Activity representerar en enskild skärm med ett användargränssnitt och en Serivce är en process som körs i bakgrunden av applikationen. De olika delarna av en applikation kan kommunicera med till exempel Intent, vilket är meddelanden med ett specifikt format som används för att till exempel byta Activity och för att överföra data mellan komponenter (Google Inc. u.å.b).

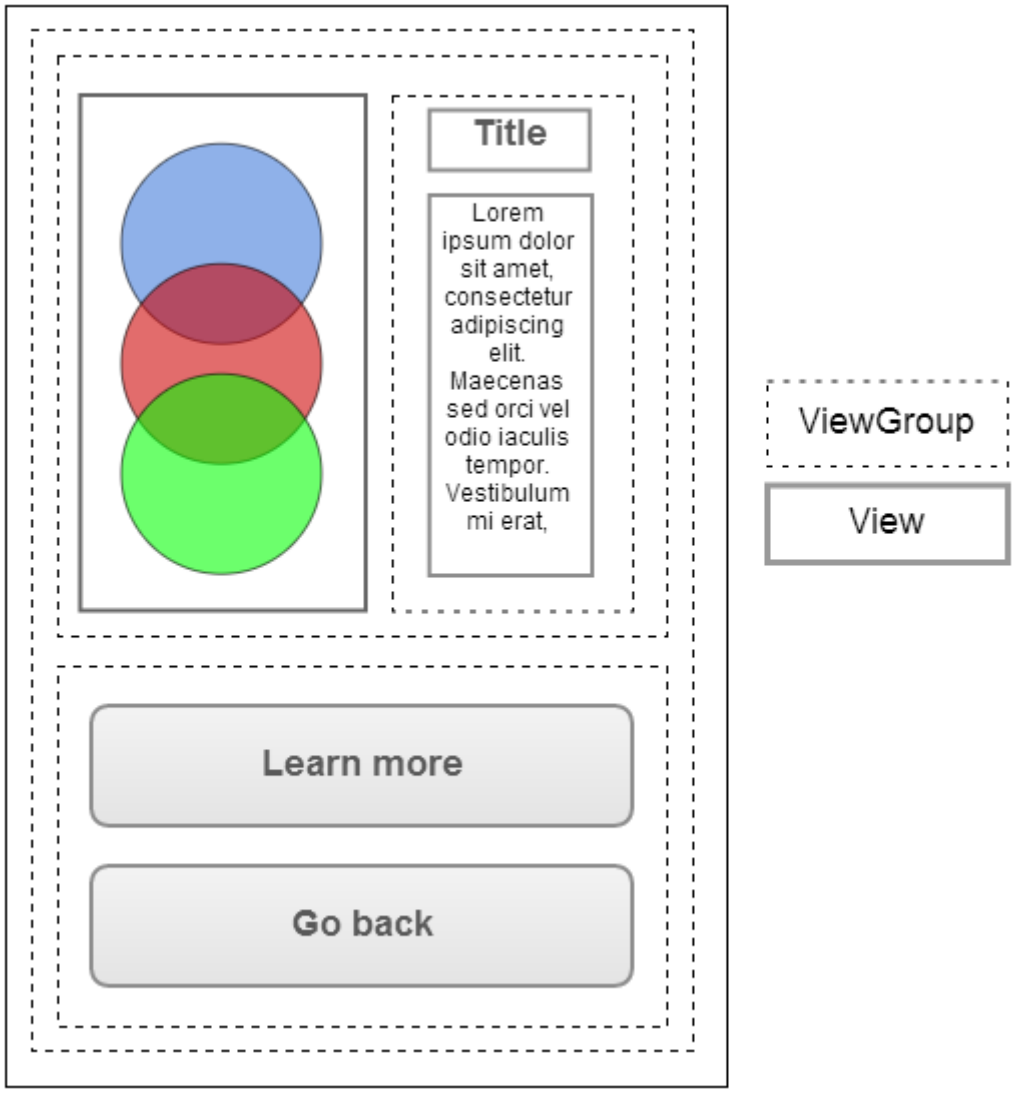
Android-komponenten Activity kommer i följande delar av rapporten att benämnas som Aktivitet.

4.3.1 Användargränsnittet

Användargränssnittet i en Aktivitet i Android byggs upp av grafiska komponenter som kallas för Views. En View kan till exempel vara en knapp, lista eller ett textfält och genom att kombinera dessa komponenter byggs användargränssnittet upp. För att organisera hur dessa komponenter skall placeras på skärmen används så kallade ViewGroups som är behållare som anger hur innehållande Views ska vara placerade i gränssnittet (Google Inc. u.å.c).

Android-komponenterna View och ViewGroup kommer i följande delar av rapporten att benämnas som Vy respektive Vygrupp.

Figur 2 visar ett exempel på hur ett användargränssnitt i en Aktivitet kan vara uppbyggt av olika Vygrupper och Vyer.



Figur 2: Ett exempel på hur ett enkelt användargränssnitt för Android kan se ut. Bilden visar också gränssnittets uppbyggnad av Vyer och Vygrupper.

4.3.2 Användarinmatning

När användaren gör någon form av inmatning till en applikation, till exempel trycker på en knapp, utför en gest eller trycker på det virtuella tangentbordet, genereras en inmatningshändelse, en så kallad InputEvent. Denna händelse är kopplad till den Vy eller Vygrupp som användaren interagerar med och ger information om var, när, hur inmatningen skedde. När en inmatningshändelse inträffar anropas fördefinierade metoder för att hantera en händelse av just denna typ. Dessa metoder finns definierade i Android-klasserna Activity och View och går att överskugga med egna definitioner (Google Inc. u.å.d).

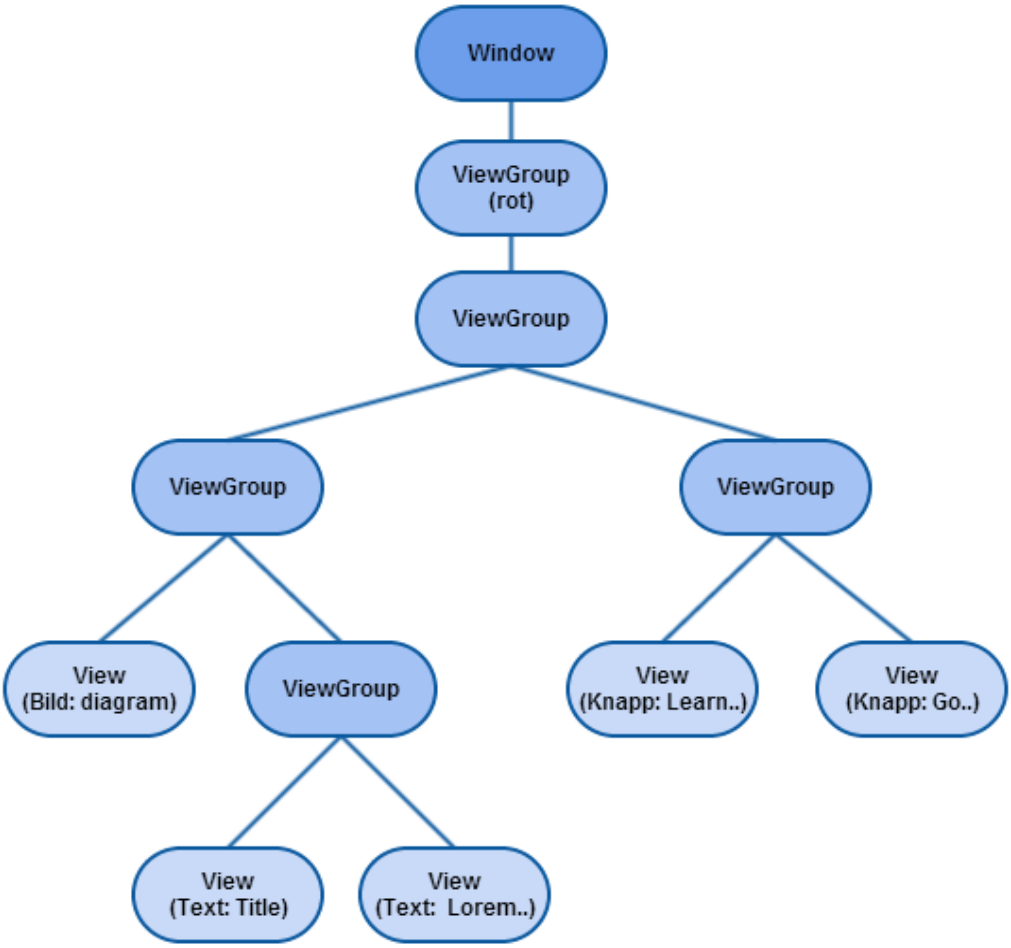
Det finns även möjligheten att hantera inmatningshändelser genom att använda händelselyssnare, så kallade EventListeners. Detta är ett Interface

med endast en metod vilken har i uppgift att hantera händelser. Dessa lyssnare kopplas till Vyer, Vygrupper eller Aktiviteter och blir anropade när händelser för dessa komponenter inträffar (Google Inc. u.å.d). Till exempel kan en knapp ha en lyssnare för tryck kopplad till sig och då anropas metoden i denna varje gång en användare trycker på knappen.

Gester och tryck på enhetens skärm representeras av en sekvens av MotionEvent-objekt av olika typ. När gesten påbörjas genereras en händelse av typen DOWN, därefter genereras händelser av typen MOVE när förflyttning sker och en händelse av typen UP när gesten avslutas. Information om vilken typ av händelse som har inträffat finns att tillgå via MotionEvent-objektet (Google Inc. 2014b).

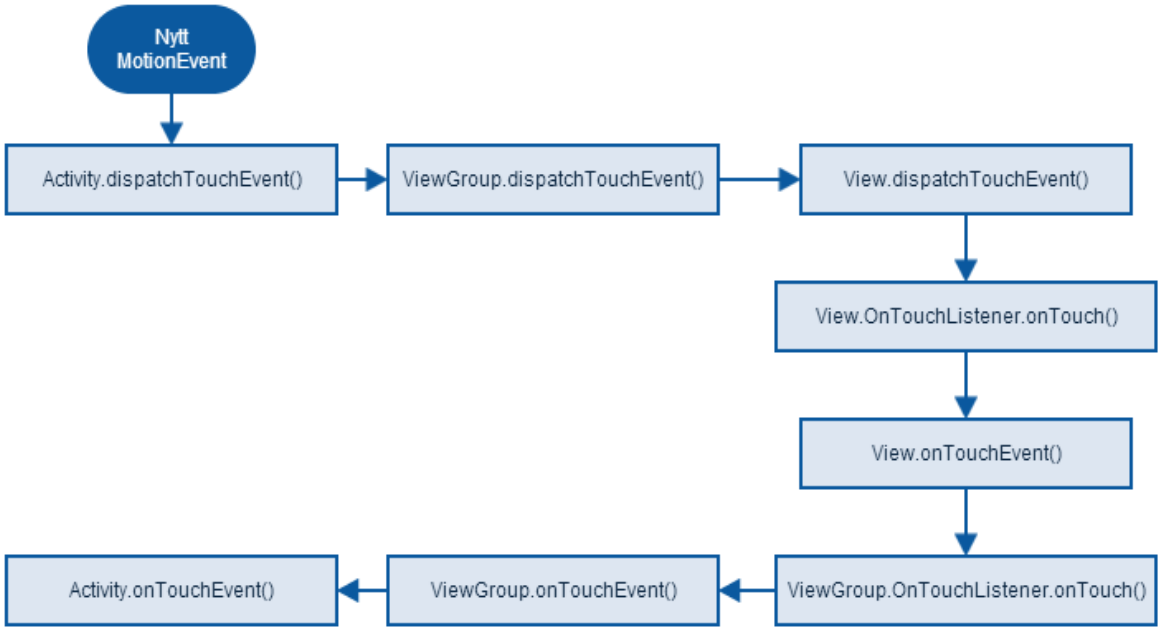
Vissa inmatningshändelser som till exempel MotionEvents kan ha flera möjliga mål, ett tryck i en lista kan till exempel vara antingen ett tryck på en enskild komponent i listan eller början till en scrollning i listan. I Android hanteras detta genom att denna typ av händelse propagerar neråt i Vy-hierarkin från förälder till barn, alltså från Vygrupp till Vy (Smith, D. 2013).

I figur 3 illustreras Vy-hierarkin för användargränssnittet i figur 2. Som figuren visar är hierarkin i form av ett träd där trädets rot är kopplad till enhetens skärm. När en ny inmatningshändelse inträffar skickas den först till den Vygrupp som är rot i trädet. Därefter skickas den vidare nedåt i trädet till alla Vyer och Vygrupper som skulle kunna vara intresserade av händelsen. Varje Vy och Vygrupp har möjligheten att konsumera händelsen och på så sätt hindra att den skickas vidare i trädet. I denna process får lyssnare tillgång till inmatningshändelser innan metoder för händelsehantering i Vyer, Vygrupper och Aktiviteter får det.



Figur 3: Vy-hierarkin för användargränssnittet i figur 2.

Figur 4 visar hur en ny skärmgest hanteras genom att en sekvens av olika metoder anropas för att avgöra var gesten skall hanteras. Först når den nya inmatningshändelsen den nuvarande Aktiviteten som vidarebefordrar denna till den Vygrupp som är rot i Vy-trädet. Roten skickar sedan vidare händelsen till alla sina barn som befinner sig inom tryckområdet, dessa barn kan vara Vyer eller Vygrupper. Vygrupper längre ned i trädet upprepar samma process som roten och de barn som är Vyer får chansen att konsumera händelsen. Om ingen Vy konsumerar händelsen propagerar händelsen tillbaka uppåt i trädet och Vygrupperna får chansen att konsumera händelsen. För både Vyer och Vygrupper hanteras inmatningshändelser i första hand av eventuella lyssnare och i andra hand av egna metoder för händelsehantering. Om ingen Vygrupp eller Vy väljer att hantera gesten hamnar den i Aktivitetens egna händelsehantering (Smith, D. 2013).



Figur 4: Ordningen metoder för händelsehantering exekveras när ett nytt skärmtryck har inträffat i en Aktivitet. Figur inspirerad av Smith, D. (2013).

Om en händelsehanterare väljer att konsumera den första inmatningshändelsen (DOWN) för en gest, kommer alla efterföljande händelser (MOVE och UP) för gesten att levereras till denna hanterare. Även om ingen annan händelsehanterare har chansen att ta över hanteringen av gesten så kommer efterföljande MotionEvents att levereras genom samma process illustrerad i figur 4. Det finns dock ett undantag, om ingen komponent i Vy-hierarkin väljer att konsumera den första händelsen för gesten och den hanteras av Aktivitetens egna händelsehanterare (Smith, D. 2013). I detta fall kommer efterföljande inmatningshändelser kopplade till gesten att levereras direkt till Aktivitetens hanterare utan att propagera genom Vy-hierarkin.

Android-applikationer exekveras som standard i en process med en tråd, om inga andra processer eller trådar skapats explicit. Denna tråd är den så kallade UI-tråden och det är i denna tråd alla gränsnittsoperationer som till exempel uppritning och händelsehantering utförs. När användarinmatning inträffar är det alltid i denna tråden som de metoder som hanterar detta exekveras. En hög belastning på denna tråd kan skapa fördröjningar i gränssnittet som användaren uppfattar som att applikationen låser sig. Därför är det viktigt att UI-tråden inte överbelastas eller blockeras av en annan tråd (Google Inc. u.å.e).

4.3.3 Förstudie: Möjligheter att mäta beteenden i Android

Ett av projektets förutsättningar var att det finns tillräckliga tekniska möjligheter att mäta användarens interaktion med en mobilapplikation. För att säkerställa att projektet var genomförbart genomfördes en förstudie för att undersöka vilka tekniska möjligheter som finns att mäta användares beteende i en befintlig Android-applikation.

Förstudien genomfördes genom att utveckla ett enkelt mätverktyg för att mäta en mängd relevanta beteendeparametrar i en enkel testapplikation.

4.3.3.1 Relevanta beteendeparametrar

Målet med förstudien var att ta fram en mängd parametrar som skulle kunna vara relevanta att mäta för att dra slutsatser om användarens beteende. Om tillräckligt många av dessa parametrar var tekniskt möjliga att mäta i en mobilapplikation fanns det förutsättningar att genomföra projektet.

De relevanta parametrarna var följande:

- Tid som en skärm är aktiv
- Scrollhastighet
- Scrollfrekvens
- Antal tryck på skärm
- Frekvens av tryck på skärm
- Tryckarea
- Varaktighet av tryck och gester
- Tryck på specifika komponenter
- Skrivhastighet i textfält
- Raderingsfrekvens i textfält
- Frekvens av skiften mellan porträtt- och landskapsläge
- Hårdhet av tryck genom accelerometer-data
- Ryck i telefon eller surfplatta genom accelerometer-data
- Tryck på tillbakaknapp
- Frekvens av olika gester
- Tid för användningssession
- Hur ofta applikation stängs och återupptas

4.3.3.2 Testapplikation

För att kunna undersöka om ovanstående parametrar är mätbara behövdes en testapplikation där samtliga förekommer. Därför utvecklades en enkel anteckningsapplikation där det går att skapa, läsa och redigera anteckningar. Denna applikation innehöll alla nödvändiga element för undersökningen, som till exempel textfält, scroll-lista och knappar.

4.3.3.3 Prototyp av mätmodul

För att testa att mäta de valda parametrarna i testapplikationen utvecklades en mätmodul som enkelt kunde implementeras utan att göra några större ändringar i befintlig kod. Målet med denna mätmodul var att registrera mätvärden från applikationen och visa dessa genom en utskrift. Modulen reagerar på händelser från gränssnittet, som till exempel tryck på en knapp eller scrollning i en lista, och mäter tillgängliga parametrar.

4.3.3.4 Resultat

Resultatet av förstudien visade att det fanns goda möjligheter att mäta beteendeparametrar i en Android-applikation. Det visade sig vara tekniskt möjligt att mäta samtliga fördefinierade parametrar. Dock visade sig problem med prestanda och integration som behövde undersökas vidare i projektet. Det upptäcktes utmaningar med att bygga en mätmodul som kan mäta samtliga parametrar i en befintlig Android-applikation och som samtidigt är enkel att integrera. Det uppkom även prestandaproblem med fördröjda svarstider i gränssnittet till följd av mätningarna. Detta var implementationsproblem som behövde undersökas och hanteras inför implementationen av den slutgiltiga prototypen.

Androids system för hantering av skärmtryck innehåller information om tryckhårdhet. Undersökningar av detta värde under förstudien visade att dess karakteristik varierar beroende av vilken hårdvaruenhet som används. Det observerades också att värdet ibland tycktes vara mer relaterat till tryckytan än den faktiska hårdheten av skärmtrycket. En lätt tryckning med hela tummen mot skärmen gav ett större värde än en hård tryckning med toppen av lillfingret.

Förstudien visade att det finns goda tekniska möjligheter att samla in mätdata om användarens beteende i Android-applikationer. Därmed fanns förutsättningar att genomföra projektet med Android som utvecklingsplattform.

4.3.4 Sammanfattning

Genom att utnyttja Androids system med InputEvents som genereras när användaren interagerar med olika delar av en applikation finns det förutsättningar att mäta användarens beteenden. Genom att lyssna på olika typer av händelser och spara data om dessa går det att kartlägga en användares beteende. Detta går att göra genom att utnyttja Androids befintliga struktur för händelsehantering med lyssnare och överskuggning av händelsehanteringsmetoder i Vyer, Vygrupper och Aktiviteter. Genom dessa

strukturer går det att få fram en stor mängd data om händelsen som kan användas för att analysera användarens beteende.

Förstudien visade att alla testade beteendeparametrar var möjliga att mäta i en Android-applikation. Det finns en stor mängd data att tillgå om användarens interaktion med en applikation i Android, så möjligheterna att mäta blev inte en begränsning för projektet. Utmaningen i Android som visade sig i förstudien låg istället i att skapa ett flexibelt och effektivt system.

Eftersom det var viktigt att det ska vara lätt att integrera *EmoTouch Android SDK* i en befintlig Android-applikation skulle det inte krävas några större ändringar av den befintliga applikationens källkod. För att uppnå detta är det möjligt att utnyttja det faktum att inmatningshändelser propagerar genom Vy-hierarkin från förälder till barn. Genom att fånga upp en händelse i denna kedja innan den når sitt mål finns det en möjlighet att mäta beteendedata utan att påverka applikationens ordinarie händelsehantering.

Ett problem som visade sig i förstudien var att gränssnittets svarstid påverkades av det extra lagret händelsehantering som modulen innebar. För att kunna mäta beteendedata krävdes det att händelser loggades och att vissa beräkningar genomfördes innan en inmatningshändelse nådde sitt ordinarie mål. Det ledde till att till exempel svarstiden för ett knapptryck blev längre vilket försämrade användarupplevelsen av applikationen. Det fanns möjlighet att förhindra detta om belastningen på UI-tråden i applikationen minskade genom att låta en separat tråd genomföra dessa extra operationer.

5 Målbild

5.1 Val av beteendeparametrar

För att *EmoTouch* ska kunna detektera beteenden som kan indikera frustration hos användare av Android-enheter behövdes ett antal parametrar definieras som är både möjliga att mäta och relevanta i förhållande till projektets mål: att dra slutsatser om en användares emotionella reaktioner och att upptäcka ett simulerat frustrerat tillstånd hos en användare. Sex stycken sådana parametrar valdes, baserat på vad som visades möjligt att mäta i förstudien (avsnitt 4.3.3) och vad som visades relevant att mäta i de studier som presenterades under avsnittet om relaterad forskning (avsnitt 4.2).

Dessa listas här i ordning efter signifikans där nummer ett anses mest signifikant.

1. Tryckhårdhet eller tryckyta - Gester och skärmtryck som utförs med högre tryck eller större tryckyta.
2. Längd av gester - Gester med kortare längd. Med längd avses det totala avståndet som användarens finger (eller penna eller annat hjälpmedel) förflyttas över skärmen från det ögonblick kontakt med skärmen uppstår tills kontakt upphör. Om användaren exempelvis utför en cirkelrörelse på skärmen definieras längden av gesten som cirkelns omkrets.
3. Hastighet av gester - Gester med högre hastighet. Med hastighet avses längden, enligt ovanstående definition, dividerat med varaktigheten av gesten.
4. Raderingsfrekvens - Högre frekvens av radering av text vid skrivning i textfält.
5. Frekvens av skärmtryck och gester - Högre frekvens av skärmtryck och gester.
6. Varaktighet av skärmtryck - Kortare varaktighet av skärmtryck. Med varaktighet avses tiden från det ögonblick kontakt med skärmen uppstår tills kontakt upphör.

En parameter som den tidigare forskningen lyfte fram som intressant men som inte är inkluderad i målbilden är skrivhastighet. Av de beteenden som Tsihrintzis et al.(2008) fann ha en koppling till ilska var det denna som ansågs ha den svagaste kopplingen. Utöver detta så ansågs det att mätning av skrivhastighet skulle vara tidskrävande att implementera i förhållande till dess signifikans. För att prioritera mer relevanta beteenden inkluderas inte skrivhastighet i målbilden.

Anledningen till att tryckhårdhet valdes som mest signfikant är att det går att mäta i nästan alla tänkbara applikationer. Parametrarna 2 och 3 går bara att mäta om applikationen innehåller scrollbara grafiska komponenter eller

detektering av olika gester, exempelvis swipe. Parameter 4 kräver att applikationen innehåller textinmatning.

Tryckhårdhet och tryckyta har kombinerats eftersom förstudien visade att Androids mätvärden av tryckhårdhet tycks påverkas av tryckyta. Även Gao, Bianchi-Berthouze och Meng (2012) menade att det går att härleda tryckhårdhet utifrån tryckyta. I följande del av rapporten kommer ordet tryckhårdhet användas för det värde Androids hantering av användarinmatning ger. Detta tycks, som tidigare nämnts, vara en kombination av area och hårdhet av tryck.

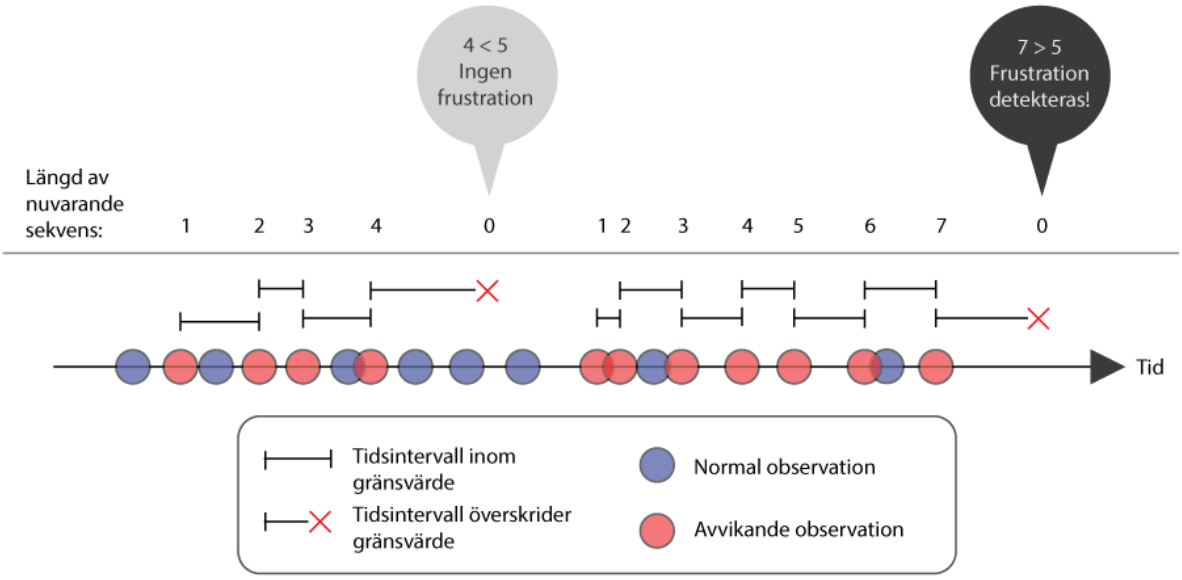
Parameter 6 kräver en kort förklaring. Resultatet av en undersökning visade ett samband mellan frustration och kort varaktighet av gester och skärmtryck. En annan undersökning visade ett samband mellan frustration och gester med hög hastighet. Eftersom hög hastighet oftast innebär kort varaktighet är det överflödigt att mäta både hastighet och varaktighet av gester. Därför mäts varaktigheten av skärmtryck och hastighet av gester som separata parametrar.

5.2 Identifiering av normalt respektive avvikande beteende

Detektion av avvikande användarbeteenden görs genom att ny indata från gränssnittet jämförs med en normalprofil för användaren. Detta görs genom att mäta ovanstående parametrar och över tid etablera ett normalbeteende för varje enskild parameter. Kombinationen av dessa parameterprofiler bildar tillsammans en normalprofil för användaren som används för att detektera avvikande beteenden.

Denna process består av två faser: en inlärningsfas och en detektionsfas. I inlärningsfasen samlas mätvärden för parametrar in och används för att etablera en normalfördelning för varje enskild parameter. När en signifikant mängd data har samlats in för en parameter inleds detektionsfasen, i vilken mätvärden jämförs med den aktuella normalfördelningen för att detektera eventuella avvikelser. I denna fas uppdateras normalprofilen med ny mätdata precis som i föregående fas. Genom att kombinera data om hur alla mätparametrar beter sig detekteras avvikande beteendemönster.

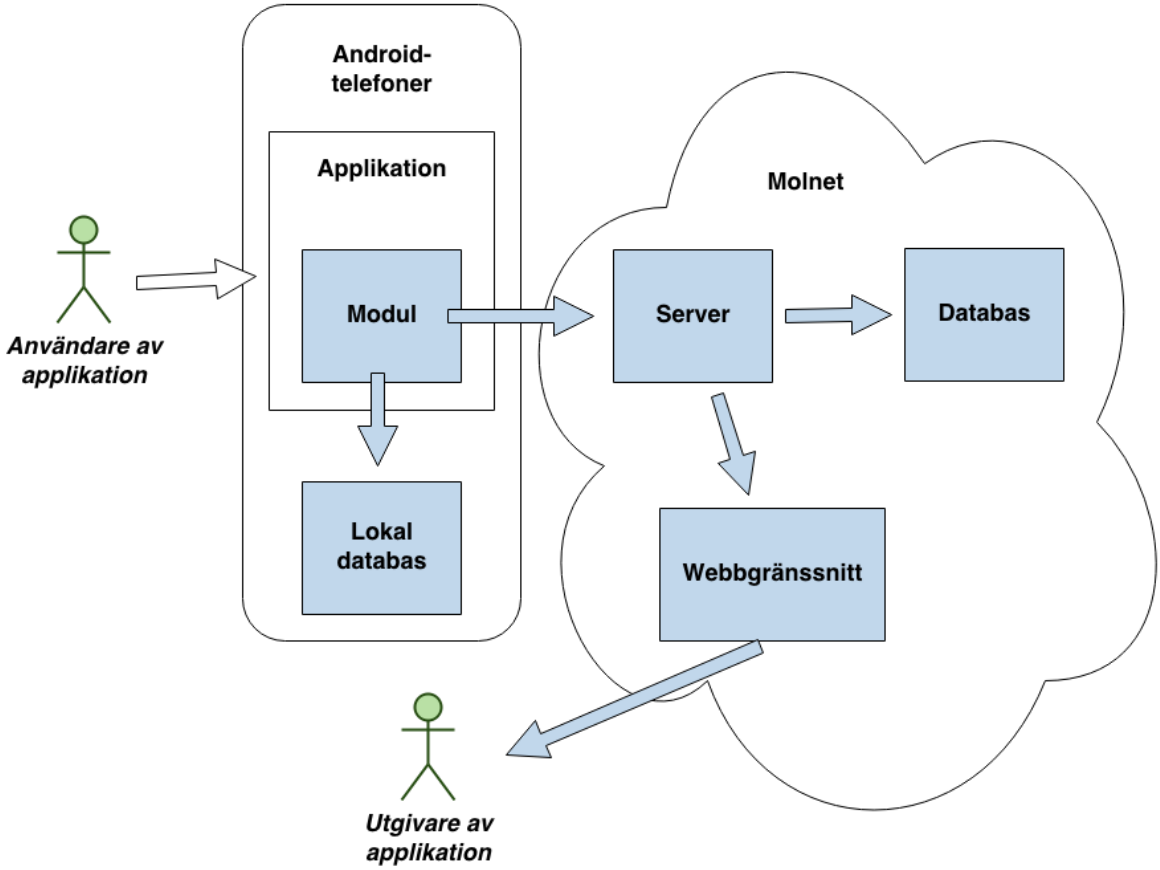
En enstaka avvikande observation är ingen tillförlitlig indikation på en möjlig avvikelse i användarens beteende. *EmoTouch* definierar därför avvikande beteende som en sekvens av avvikande observationer, där tiden mellan två efterföljande observationer inte får överstiga ett visst gränsvärde, och längden av sekvensen måste överstiga ett annat gränsvärde. Gränsvärdena är statiska. Detta illustreras av figur 5.



Figur 5: Illustration av hur EmoTouch detekterar avvikande beteende utifrån avvikande observationer. När en sekvens tar slut detekteras frustration om längden av sekvensen överskrider ett statiskt gränsvärde, som i det här exemplet valts till 5.

5.3 EmoTouch

EmoTouch består av tre delar, en Android-modul, en server och en webbapplikation. Android-modulen är ett utvecklingsbibliotek som enkelt kan integreras av en utvecklare i en befintlig Android-applikation. Servern har som funktion att ta emot information från modulerna i olika användares enheter och lagra denna. Webbapplikation hämtar data från servern och presenterar information om användarnas beteende för applikationens utgivare. Kontextdiagrammet i figur 6 visar systemets uppbyggnad och hur de olika delarna är kopplade till varandra.

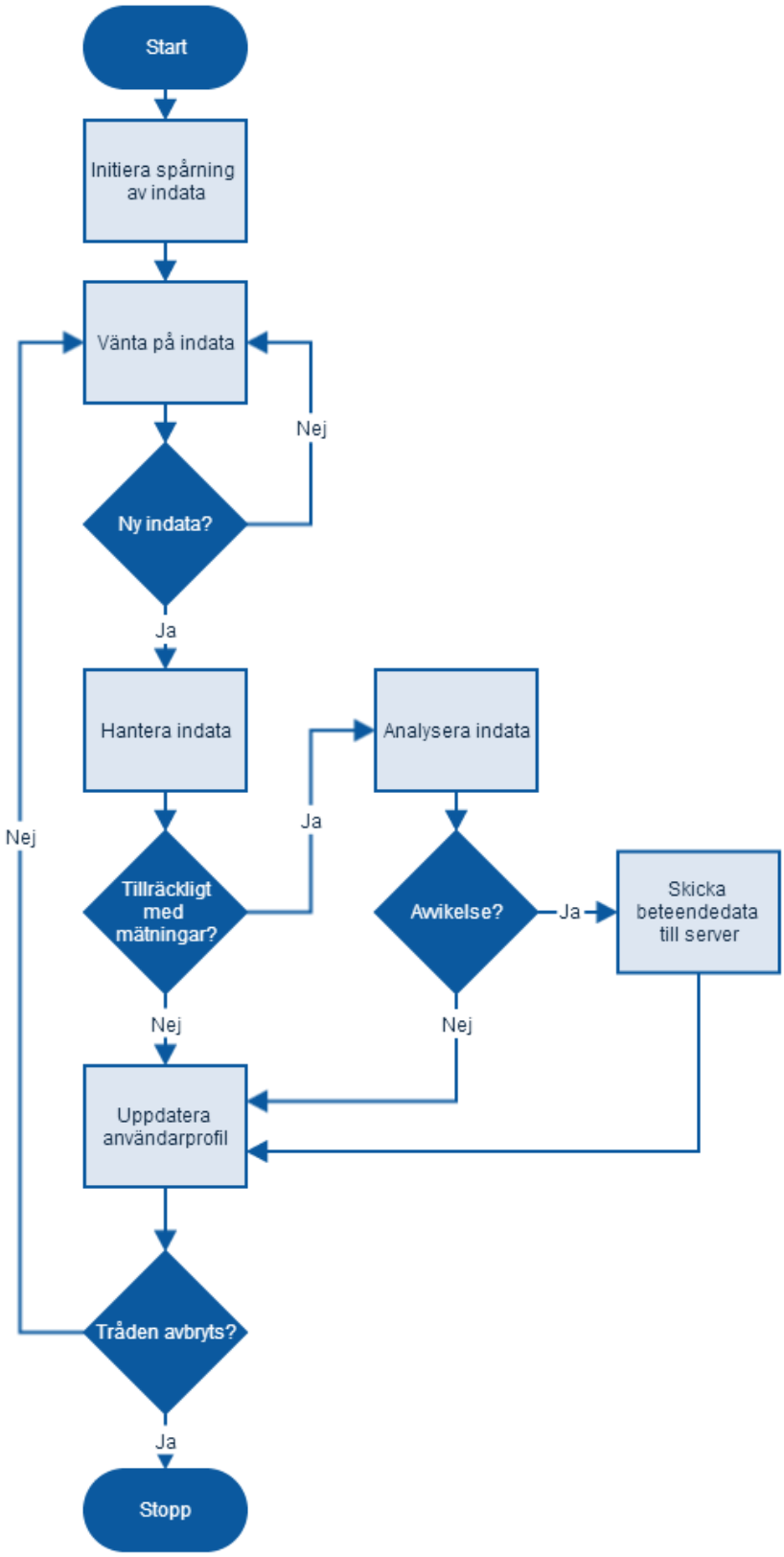


Figur 6: Kontextdiagram för EmoTouch. Pilarna visar de tekniska gränssnitten mellan olika delsystem. EmoTouch utgörs av de delar som är blåfärgade.

5.3.1 EmoTouch Android SDK

Uppgifterna för *EmoTouch Android SDK* är att mäta valda beteendeparametrar och analysera dessa för att upptäcka avvikelser. Detta görs genom att modulen kopplar in sig på applikationens händelsehantering. Modulen bygger och hanterar en normalprofil utifrån parametrars mätvärden. Eventuell detektion av avvikande beteenden rapporteras till en central server. Denna funktionalitet illustreras av figur 7.

Utöver beteendedata rapporterar modulen meta-data om användaren till servern, som till exempel operatör, geografisk position eller information om enheten. *EmoTouch Android SDK* erbjuder möjligheten för utvecklaren att lägga in egendefinierade händelser på valfria ställen i källkoden, så att ett meddelande skickas till servern vid en viss händelse i applikationen. Detta kan exempelvis vara vid ett knapptryck. Dessa egendefinierade händelser kommer att kallas *EmoTouch*-händelser i de följande delarna av rapporten.



Figur 7: Översiktligt flödesschema för EmoTouch Android SDK.

5.3.2 EmoTouch Server
Serverns funktion är att samla in data om applikationanvändarnas beteenden och lagra denna. Detta görs genom att instanser av *EmoTouch Android SDK* meddelar servern när en detektion av avvikande beteende har inträffat. Utöver att samla in och lagra data gör servern denna data tillgänglig för webbapplikationen *EmoTouch Dashboard*.

5.3.3 EmoTouch Dashboard
EmoTouch Dashboard hämtar data om alla applikationsanvändare från servern och presenterar detta i form av grafer. Detta skall användas för att övervaka beteendet av alla aktiva applikationsanvändare vid det aktuella tillfället. Data i *EmoTouch Dashboard* uppdateras i realtid samtidigt som instanser av modulen rapporterar in nya detektioner.

EmoTouch Dashboard är en webbapplikation som syftar till att demonstrera systemets funktionalitet och olika möjliga användningsområden, samt att undersöka möjligheter att visualisera realtidsdata från servern. Det går via webbapplikationen att ta fram både realtidsdata och översiktsdata över applikationsanvändning.

Följande realtidsdata är tillgänglig:
- Antal aktiva användare just nu.
- Antal detektioner av frustration just nu.
- Andel frustrerade användare just nu fördelade över länder.

Följande översiktsdata är tillgänglig:
- Total andel frustrerade användare.
- Andel frustrerade användare fördelade över olika mobila enheter.
- Andel frustrerade användare fördelade över olika städer.
- Andel frustrerade användare fördelade över olika skärmar i applikationen.

6 Fas 2

6.1 Tekniska verktyg och kodbibliotek

För att implementera *EmoTouch* har vissa tekniska verktyg och kodbibliotek använts för att möjligöra delar av systemets funktionalitet och för att förenkla utvecklingsarbetet. Nedan följer en beskrivning av de viktigaste verktyg och kodbibliotek som har använts i projektet.

6.1.1 Google Analytics

Google Analytics för Android var inte bara relevant för det här projektet som ett exempel på en liknande tjänst, utan kan också användas som ett verktyg för att hantera *EmoTouchs* server-funktionalitet. Dess Android SDK tillåter utvecklaren att skicka meddelanden vid utvalda händelser i applikationen, vilka innehåller fyra datafält som kan användas för att överföra valfri information om händelsen. Dessa meddelanden kommer att kallas Google Analytics-händelser hädanefter i rapporten. Google tillhandahåller REST API för att hämta ut denna information tillsammans med annan data såsom antalet aktiva användare vid tidpunkten för händelsen, deras geografiska position och vilken modell av enhet de använder (Google Inc. 2014c).

6.1.1.1 Google Analytics REST API

Google Analytics erbjuder fem olika REST API för att utvecklare ska kunna bygga egna applikationer som visualiserar data från Analytics. Utvecklaren konstruerar förfrågningar bestående av ett antal fördefinierade datafält och Googles server svarar med en datatabell innehållandes den eftersökta informationen. För att underlätta för utvecklare vid hanteringen av förfrågningar och svar tillhandahåller Google kodbibliotek för olika programmeringsspråk.

Core Reporting API (v3) är ett av dessa REST API. Det tillåter utvecklare att konstruera förfrågningar som definierar vilka datafält som ska hämtas, samt hur svaren ska filtreras och sorteras. Det går till exempel att begära antalet aktiva användare per minut den senaste veckan, eller antalet Google Analytics-händelser av en viss typ som har skickats. Google erbjuder bland annat ett kodbibliotek för Core Reporting API (v3) för Javascript-applikationer som begär data asynkront via AJAX-anrop. Core Reporting API (v3) är inte avsett för att hämta realtidsdata, och Google ger inga garantier om hur lång tid det tar för datan att uppdateras. Om en ny användare startar sin Android-applikation vid en viss tidpunkt är det alltså osäkert hur lång tid det dröjer innan en begäran efter antalet aktiva användare vid den tidpunkten kommer att ge det korrekta antalet. (Google Inc. 2014f).

Real Time Reporting API (v3) är ett annat REST API som i dagsläget är i beta-stadie och till stor del fungerar likadant som Core Reporting API (v3). Skillnaderna är att Real Time Reporting API (v3) är avsett för att hämta realtidsdata och är mer begränsat när det gäller vilka datafält som får begäras. Google ger dock inga garantier för hur lång tid det tar för data att uppdateras. Enligt dokumentationen kommer datan i svaret representera situationen "just nu", men detta är ett relativt begrepp. (Google Inc. 2014g).

6.1.2 OAuth 2.0 Protocol

OAuth 2.0 är ett protokoll som används för att autentisera och auktorisera en tredje part att använda en HTTP-tjänst. Genom protokollet kan en tredje part, till exempel en webbapplikation, få begränsad tillgång till resurser kopplade till en användares konto hos en tjänsteleverantör. Detta görs genom att användaren genom OAuth 2.0 ger den tredje parten tillgång till vissa resurser genom att autentisera sig hos tjänsteleverantören. Den tredje parten blir då tilldelad en nyckel som under begränsad tid ger tillgång till en angiven del av användarens resurser. (Hardt, D. 2012)

Google använder OAuth 2.0 protokollet för att tredje parter skall kunna få tillgång till deras API för att använda resurser kopplade till användarkonton. Till exempel går det att använda OAuth 2.0 för att hämta data från ett Google Analytics-konto. (Google Inc. 2014h)

6.1.3 jQuery v1.11.0

jQuery är ett JavaScript-bibliotek med en mängd funktioner för manipulation av HTML-dokument, hantera användarinmatning och AJAX-anrop. Biblioteket förenklar utvecklingen av webbapplikationer genom att ha inbyggt stöd för följande webbläsare: Internet Explorer, Chrome, Firefox, Safari, Opera, iOS och Android. (The jQuery Foundation 2014)

6.1.4 Bootstrap v3.1.1

Bootstrap är ett CSS- och JavaScript-bibliotek som ursprungligen är utvecklat av Twitter Inc. och förenklar utvecklingen av webbsidor genom att erbjuda en stor mängd designmallar. Biblioteket innehåller en mängd komponenter byggda i HTML, CSS och JavaScript som kan användas som byggstenar för webbsidor. Bootstrap förenklar även utvecklingen av webbsidor som ska användas från flera olika typer av enheter (till exempel mobiler, surfplattor och persondatorer). (Otto, M. & Thornton, J. 2014)

6.1.5 Highcharts v4.0.1

Highcharts är ett JavaScript-bibiolitek som erbjuder komponenter för interaktiv datavisualisering i form av grafer och kartor. Biblioteket innehåller en stor mängd olika typer av grafer som till exempel stapeldiagram, areadiagram, linjediagram och geodiagram. Alla grafer är interaktiva och kan reagera på användarinmatning (till exempel musklick och scroll). Datan som visualiseras går att uppdatera dynamiskt efter det att grafen har skapats. Highcharts är fritt att använda i icke-kommersiella produkter, för kommersiella produkter krävs en licens. (Highsoft AS. 2014)

6.2 Systemets implementation

Under projektets andra fas har ett system motsvarande det beskrivet i målbilden utvecklats. Det utvecklade systemet *EmoTouch* består av två delar: En Android-modul med namnet *EmoTouch Android SDK* och en webbapplikation med namnet *EmoTouch Dashboard* för att visualisera den insamlade datan.

Det följande kapitlet innehåller en redogörelse för utvecklingsarbetet och en beskrivning av slutresultatet.

6.2.1 EmoTouch Server: Google Analytics

EmoTouch använder Google Analytics som verktyg för all backend-funktionalitet. Detta innebär att ingen serverapplikation implementerades under projektet. *EmoTouch Android SDK* använder *Google Analytics Android SDK* för att skicka meddelanden innehållandes information till Googles server. Informationen skickas via Google Analytics-händelser. *EmoTouch Dashboard* använder Google Analytics REST API för att hämta informationen och presentera den grafiskt.

Valet att använda Google Analytics motiveras av:

- *EmoTouch* bygger vidare på den funktionalitet Google Analytics erbjuder för Android-applikationer. Därför är det rimligt att *EmoTouch* konstrueras som ett tillägg eller en plugin till Google Analytics.
- Google Analytics erbjuder en stabil plattform för kommunikation mellan *EmoTouch Android SDK* och server, mellan server och *EmoTouch Dashboard*, samt datalagring på serversidan.
- Utöver den information *EmoTouch Android SDK* sammanställer, samlar Google Analytics Android SDK in annan data som kan vara relevant för att en applikationsutgivare ska kunna dra slutsatser om orsaken till användares frustration, såsom användarens geografiska position, modell av mobil enhet etc.

- Det ger *EmoTouch* en större trovärdighet ur applikationsutgivares perspektiv, eftersom Google Analytics är ett välkänt, etablerat verktyg. Applikationsutgivaren kommer då att kunna använda sitt Google Analytics-konto för att autentisera sig vid användning av *EmoTouch Dashboard*. Ansvaret för att information om användarna behandlas på ett säkert sätt ligger hos Google. Detta argument är egentligen inte relevant vid utvecklandet av en prototyp, men är värt att väga in om *EmoTouch* vidareutvecklas för lansering på marknaden.

Användandet av Google Analytics innebar dock vissa komplikationer. Google Analytics REST API erbjuder ingen möjlighet att koppla Google Analytics-händelser till en viss användare eller ett visst användningstillfälle (Google Inc. 2014c). Detta begränsar vilken information *EmoTouch Dashboard* kan presentera. Det går till exempel att ta reda på hur många gånger under en tidsperiod någon användare uppvisat avvikande beteende, men inte hur många unika användare som uppvisat avvikande beteende under en tidsperiod. Är det 100 användare som varit frustrerade varsin gång eller 10 användare som varit frustrerade 10 gånger var? Detta har lösts genom att *EmoTouch Android SDK* har implementerats på ett sådant sätt att den väntar ett visst tidsintervall efter att den notifierat servern om avvikande beteende, innan den kan notifiera nästa gång. Då blir antalet frustrerade händelser lika med antalet frustrerade användare om man tittar på datan i realtid, eftersom samma användare inte kan utlösa flera händelser.

En annan konsekvens av att det inte går att koppla Google Analytics-händelser till användare eller användningstillfälle är att det inte går att koppla användarens detekterade frustration till en *EmoTouch*-händelse. Det kan exempelvis vara intressant för en applikationsutgivare att se fördelningen av frustrerade användare i samband med att användarna använder en viss funktion i applikationen. För att åstadkomma detta måste *EmoTouch* skapa en unik kod för varje användningstillfälle som kan skickas med varje meddelande till servern och användas för att koppla olika meddelanden till samma användningstillfälle. Detta kan implementeras genom att varje användare får en unik identifieringskod som kan kombineras med en siffra som räknar användarens antal användningstillfällen av applikationen. Det är dock inte helt oproblematiskt att skapa sådana identifierare om systemet ska fungera för ett stort antal användare.

En möjlighet är att låta varje instans av *EmoTouch Android SDK* välja en pseudo-unik kod för användaren. Den enda nackdelen med detta är att det finns en sannolikhet att två olika användare slumpar samma kod. En annan möjlighet är att använda en befintlig kod som redan är unik för användarens enhet, såsom nätverkskortets MAC-adress. Detta är dock problematiskt

eftersom det kan anses bryta mot Googles användarvillkor för Analytics, där det föreskrivs att verktyget inte får användas för att lagra data som kan användas för att identifiera en person (Google Inc. 2014d). Enligt Banks (2012) kan en MAC-adress räknas som sådan information, men det är något av en juridisk gråzon och beroende av sammanhanget. En annan möjlighet är att låta en server ansvara för att dela ut unika identifierare. Dessa skulle då kunna göras helt unika och de hade inte kunnat klassificeras som personlig information. Nackdelen är att det hade krävt en egen server enbart för uppgiften.

EmoTouch använder en pseudo-unik kod för att koppla olika meddelanden till samma användningstillfälle. Detta implementeras med hjälp av Android-klassen UUID som erbjuder funktionalitet att skapa en sådan 128-bitars kod (Google Inc. 2014e). Sannolikheten att två olika installationer av en applikation producerar samma identifierare är mikroskopisk, och av de tre möjliga lösningarna som presenterats är detta den som är minst komplex att implementera.

6.2.2 EmoTouch Android SDK

För att kunna övervaka en applikationsanvändares beteende har *EmoTouch Android SDK* utvecklats. Modulen är konstruerad för att integreras i en befintlig Android-applikation. En Android-applikation som använder *EmoTouch* måste ha Google Analytics Android SDK integrerat och aktivt i sin applikation. Detta eftersom modulen använder sig av Google Analytics för att lagra detektioner av avvikande beteende och få tillgång till annan användardata.

En användarens beteende analyseras genom att *EmoTouch* övervakar hur användaren interagerar med enhetens skärm och matar in text i applikationen. Detta görs genom att modulen snappar upp sådana händelser innan de når själva applikationen. Om *EmoTouch* detekterar något som kan vara en indikation på ett frustrerad beteende så skickas ett meddelande till Google Analytics.

För att undvika att belasta applikationens ordinare exekvering så körs *EmoTouch* till största delen i en egen tråd. Det är i denna tråden alla beräkningar kopplade till beteendeanalysen utförs.

Här följer en redogörelse av de viktigaste valen som gjordes under implementationsarbetet med *EmoTouch Android SDK*.

6.2.2.1 Integrering av modul i Android-applikation

En av målsättningarna med *EmoTouch* var att det skulle gå att integrera modulen i en befintlig applikation på ett sätt som inte kräver mer handpåläggning från utvecklaren än vid integration av liknande verktyg (Till exempel Google Analytics, Appsee eller Countly). Enligt denna målsättning har *EmoTouch* designats på ett sådant sätt att det endast krävs två metodanrop till *EmoTouch* i varje Aktivitet för att aktivera beteendeövervakning i applikationen. Det som krävs är att *EmoTouch* anropas när en Aktivitet startas och stoppas, detta görs för att meddela *EmoTouch* att övervakning av användarbeteendet i en Aktivitet ska påbörjas respektive avslutas. Detta är samma mängd anrop som krävs för att aktivera Google Analytics övervakning av en Android-applikation. (Google Inc. 2013)

Figur 8 visar hur *EmoTouch* integreras i en Aktivitet genom att lägga till två rader kod sist i de överskuggade metoderna onStart() och onStop(). Det första anropet i metoderna startar/stoppar Google Analytics spårning av Aktiviteten och det andra startar/stoppar *EmoTouch* beteendeövervakning. Eftersom *EmoTouch* använder sig av funktionalitet i Google Analytics krävs det att detta är aktiverat för att modulen skall fungera.

```
@Override
protected void onStart() {
    super.onStart();

    EasyTracker.getInstance(this).activityStart(this);
    EmoTouch.getSharedInstance().start(this);
}

@Override
protected void onStop() {
    super.onStop();

    EasyTracker.getInstance(this).activityStop(this);
    EmoTouch.getSharedInstance().stop();
}
```

Figur 8: Kod för att integrera EmoTouch Android SDK i en Android-aktivitet.

Det är viktigt att det är lika lätt att börja använda *EmoTouch* i en applikation som det är att använda liknande verktyg. Även om *EmoTouch* är en protoyp så är det viktigt att det inte är för komplicerat att börja använda systemet för att en framtida version ska vara kommersiellt gångbar. *EmoTouch* kan anses tillräckligt enkelt att integrera eftersom integrering sker på samma sätt som med Google Analytics.

För att applikationer ska kunna reagera direkt på användares emotionella reaktioner finns möjligheten att koppla en lyssnare, OnFrustrationListener till *EmoTouch*. Koden i lyssnaren exekveras varje gång *EmoTouch* detekterar frustration och eftersom koden exekveras på UI-tråden kan utvecklaren låta gränssnittet reagera på frustrationen.
Detta skapar möjligheten att skapa applikationer som direkt reagerar på att användaren är frustrerad. Till exempel kan applikationen försöka minska användarens frustration genom att vägleda användaren mot sitt mål. Figur 9 visar ett exempel på hur en lyssnare kan läggas till för att skriva ut ett meddelande på skärmen varje gång frustration detekteras.

```
EmoTouch.getSharedInstance().setOnFrustrationListener(new OnFrustrationListener() {

    @Override
    public void onFrustration(int frustrationScore) {
        Toast.makeText(BrowsePostsActivity.this, "FRUSTRATION. score : " + frustrationScore,
                Toast.LENGTH_LONG).show();
    }
});
```

Figur 9: Kod för att koppla en lyssnare till EmoTouch som skriver ut ett meddelande på skärmen varje gång frustration har detekterats.

För att koppla indikationer på frustration till händelser i applikationen kan applikationsutgivare lägga till *EmoTouch*-händelser. Detta görs genom att anropa *EmoTouch* vid intressanta punkter under applikationens exekvering. *EmoTouch* kopplar dessa händelser till ett specifikt användningstillfälle för en specifik användare. Genom att i efterhand undersöka var under applikationens exekvering eventuell frustration har detekterats kan värdefull information om användarproblem upptäckas.

I figur 10 visas ett exempel på en metod som exekveras varje gång en användare i en E-handelsapplikation navigerar till kassan. I denna egendefinierade händelse anges även hur många föremål användaren har i sin varukorg. Genom att undersöka detektioner av frustration i samband med denna händelse kan intressant information om användarnas beteende upptäckas. Om exempelvis ovanligt många användare uppvisar ett frustrerat beteende i samband med betalning finns det antagligen problem med den skärmen som behöver undersökas närmre.

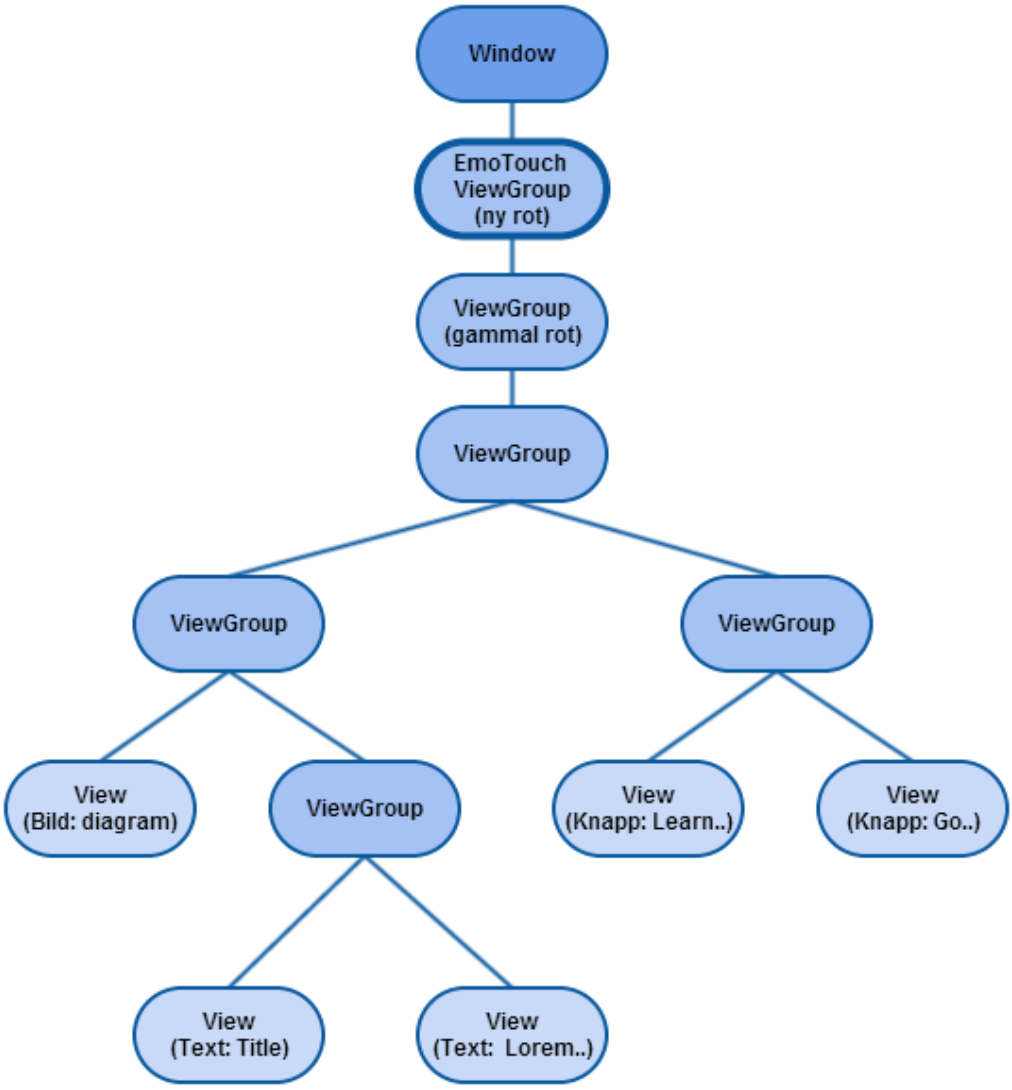
```
private void checkOut() {
    EmoTouch.getSharedInstance().sendCustomEvent("Checking out", numItemsInShoppingCart);
}
```

Figur 10: Kod för att skicka en egendefinierad händelse med EmoTouch. I detta exempel är händelsen att en användaren av en E-handelsapplikation ska betala. Händelsen innehåller även information om antalet varor i varukorgen.

6.2.2.2 Hantering av användarinmatning

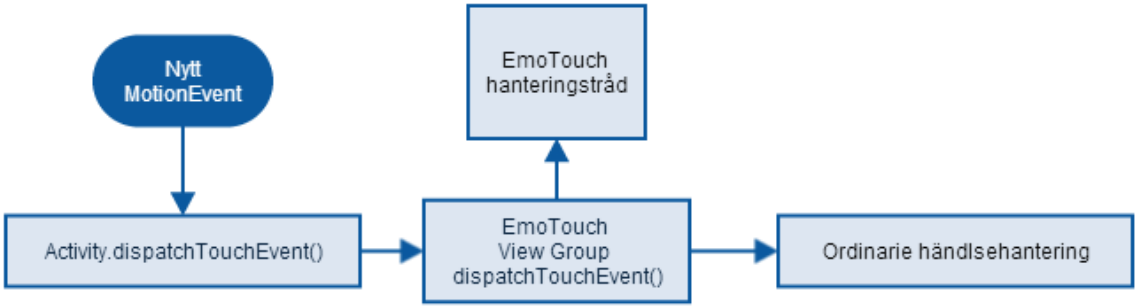
En viktig del av *EmoTouch* funktion är att fånga upp inmatningshändelser för att kunna analysera dessa. En av utmaningarna som visade sig i förstudien (avsnitt 4.3.3) var hur detta skulle göras på sätt som inte stör applikationens ordinarie händelsehantering eller kräver något större arbete från applikationsutvecklaren. I *EmoTouch* fångas inmatningshändelser upp utan att applikationsutvecklaren behöver göra några ändringar i sin händelsehantering.

Tryck och gester på en Android-enhets skärm genererar MotionEvent-objekt som innehåller en mängd information om händelsen. I *EmoTouch* fångas MotionEvents upp genom att utnyttja det faktum att händelser i Android propagerar nedåt i Vy-hierarkin. Det innebär att händelser alltid passerar roten i hierarkin innan de når sina mål och konsumeras. Genom att *EmoTouch* skapar en ny Vygrupp och kopplar på denna som ny rot i Vy-hierarkin så kan alla MotionEvents i applkationen fångas upp innan de når sitt riktiga mål. I figur 11 illustreras Vy-hierarkin för användargränssnittet i figur 2 med *EmoTouch* aktiverat.



Figur 11: Vy-hierarkin för användargränssnittet i figur 2 med EmoTouch Android SDK aktiverad.

MotionEvents avlyssnas genom att Vygrupp-metoden "dispatchTouchEvent(MotionEvent)" i den nya roten överskuggas. Denna metod anropas varje gång ett MotionEvent med ett mål i hierarkin genereras och eftersom *EmoTouch* befinner sig i roten, kommer denna metod anropas först för alla nya MotionEvents i Aktiviteten. *EmoTouch* fångar då upp händelsen innan den skickas vidare mot sin mål-Vy. I figur 12 illustreras det hur *EmoTouch* fångar upp MotionEvents och vidarebefordrar dem till en hanteringstråd innan den ordinarie händelsehanteringen fortsätter.



Figur 12: Händelsehanteringen när ett nytt MotionEvent har genererats. För den ordinarie händelsehanteringen, se figur 4.

Om ett MotionEvent inte har en intressent i Vy-hierarkin konsumerar *EmoTouch* själv händelsen för att förhindra att efterföljande händelser levereras direkt till aktivitetens egen händelsehantering. Genom att göra detta kommer händelsen att fortsätta propagera genom Vy-hierarkin och kan fångas upp av *EmoTouch*. Eftersom det är möjligt att Aktiviteten har som uppgift att hantera sådana händelser, skickar *EmoTouch* vidare dessa händelser till Aktiviteten. Detta görs för att inte rubba applikationens ordinarie händelsehantering.

Texthändelser fångas upp på ett annat sätt eftersom Android hanterar dessa händelser på ett annorlunda sätt än gester/tryck. För att fånga upp dessa händelser används istället en annan konstruktion i Android, en TextWatcher. Genom att koppla en TextWatcher till ett textfält i Android anropas denna varje gång texten i fältet förändras. För att fånga upp alla texthändelser i en Aktivitet kopplar *EmoTouch* en TextWatcher till varje textfält i Aktiviteten. Samma TextWatcher kopplas till samtliga textfält genom att iterera och undersöka alla Vyer i Aktiviteten.

6.2.2.3 Arbetstråd – Producer-Consumer med mailbox

Ett problem med mätmodulen till Android som visade sig under förstudien (avsnitt 4.3.3) var försämrad svarstid i gränssnittets när modulen var aktiv. Detta beror på att Android-applikationer som standard endast exekveras i en process med en tråd, den så kallade UI-tråden. Eftersom modulen ligger mellan gränssnittet och applikationens händelsehanteringen fördröjs gränssnittets reaktion på användarinmatning medan modulen analyserar indatan. Detta orsakar en liten men märkbart längre fördröjning mellan inmatning och att applikationen reagerar på händelsen.

För att förhindra att modulen orsakar sämre svarstiden i applikationers gränssnitt körs modulens analys av indata i en egen arbetstråd. Denna tråden bildar tillsammans med UI-tråden ett Producer-Consumer par som

kommunicerar asynkront. Trådarna kommunicerar genom att meddelanden placeras i en buffert-monitor, en så kallad mailbox. UI-tråden producerar data till arbetstråden genom att placera meddelanden med händelsedata i bufferten innan de skickas vidare till applikationens ordinarie händelsehantering. Om bufferten är full blockeras ej UI-tråden av detta. Arbetstråden försöker hämta meddelanden från bufferten och om det inte finns några blockeras tråden tills ett nytt meddelande finns att hämta. Därefter analyseras datan i meddelandet beroende på innehåll.

Målsättningen med valet av denna typ av systemdesign är att minimera modulens påverkan på applikationens prestanda. Detta görs genom att UI-tråden aldrig blockeras av arbetstråden och endast behöver utföra ett minimalt antal extra operationer för att vidarebefordra händelser till arbetstråden. Eftersom arbetstråden är blockerad när det inte finns något arbete att utföra hindrar den inte UI-tråden från att exekveras i onödan, utan använder endast enhetens CPU vid behov. Med denna uppbyggnaden av modulen noteras ingen fördröjning av gränssnittets svarstid.

6.2.2.4 Arbetsfördelning mellan server och modul

En avvägning vid designen av *EmoTouch* var hur ansvarsfördelningen mellan server och modul skulle se ut. Frågan var om modulen skulle genomföra beräkningarna och endast meddela servern vid en avvikelse eller om servern skulle genomföra beräkningarna medan modulen endast vidarebefodrar data till servern. Detta var en avvägning mellan mängden datatrafik som modulen skickar och mängden beräkningar som utförs av modulen.

I *EmoTouch* sker alla beräkningar kopplade till beteendeanalysen i modulen och servern meddelas endast vid en eventuell avvikelse. Modulen både mäter och analyserar indata från användaren. Den lagrar användarens normalprofil lokalt, därmed behöver data om normalprofilen ej hanteras på servern. Servern tar emot meddelanden om att avvikelser har inträffat och och lagrar dessa för visualisering i *EmoTouch Dashboard*.

Med denna uppbyggnad av *EmoTouch* behöver inte stora mängder data hanteras på servern, utan endast information om inträffade avvikelser. Därmed behövs det ingen komplex serverlösning med förmågan att hantera stora dataflöden. En annan fördel är att modulen ej behöver skicka stora mängder datatrafik, vilket är att föredra då många mobila enheter har en begränsad tillåten datatrafik. Eftersom normalprofilen lagras lokalt på enheten kan den uppdateras även utan tillgång till Internet.

Ett eventuellt problem med att genomföra beräkningarna för beteendeanalysen i modulen är att applikationens prestanda påverkas negativt. I *EmoTouch* används inga tunga beräkningar vid beteendeanalysen och därmed kan dessa utföras i modulen utan att någon påverkan på prestanda noterats vid testning. Om en mer beräkningstung metod, som till exempel någon form av maskinlärnings-algoritm, hade använts så är det troligt att det hade behövts mer beräkningskraft än vad de flesta mobila enheter har att erbjuda.

6.2.3 Modell för detektion av avvikande beteende

Att kartlägga en användares normalbeteenden och detektera avvikande beteenden kan göras med hjälp av olika modeller med varierande grad av komplexitet. En av de mindre komplexa metoderna bygger på antagandet att värdet av de olika beteendeparametrarna är normalfördelade. Metoden går ut på att kontinuerligt beräkna och spara ett medelvärde och en standardavvikelse för varje mätbar parameter, och för varje ny observation kontrollera om det observerade värdet ligger utanför standardavvikelsen i den riktning som anses indikera frustration. För exempelvis tryckhårdhet är det observationer som är större än medelvärdet plus standardavvikelsen som är av intresse, men för gestlängd är det observationer som är mindre än medelvärdet minus standardavvikelsen. Observationen räknas då som en del av ett avvikande beteende.

En mer komplex typ av modell är att använda någon form av maskininlärning. Detta innebär att ett datorprogram analyserar en stor mängd data enligt en avancerad algoritm och identifierar betydelsefulla mönster. Exempelvis använde Gao, Bianchi-Berthouze och Meng (2012) maskininlärningsalgoritmer för att analysera användarbeteenden i den studie som tidigare beskrivits i rapporten. De använde sig då av så kallad övervakad maskininlärning vilket innebär att det först finns en inlärningsfas då datorprogrammet får analysera en stor mängd observerade värden som har klassificerats i olika kategorier. Därefter kan programmet själv kategorisera kommande observationer utifrån de mönster som fastställts under inlärningsfasen. Det går även att göra oövervakad inlärning vilket innebär att programmet kan gruppera olika observationer utan fördefinierade kategorier (Mitchell 1997).

EmoTouch använder en modell baserad på normalfördelning av beteendeparametrar, vilken antogs kunna ge tillräcklig precision vid detektion av avvikelser. Att tillämpa maskininlärning i detta projekt hade inneburit följande svårigheter:

- Övervakad maskininlärning kräver en inlärningsfas för att avvikelser senare ska kunna detekteras. Eftersom *EmoTouch* använder en användarspecifik normalprofil hade detta inneburit att varje användare,

under inlärningsfasen, på något sätt kategoriserar sitt beteende som frustrerat eller icke-frustrerat. Detta ansågs orimligt att implementera på ett sätt som inte irriterar användaren eller stör applikationens övriga funktionalitet.

- Oövervakad maskininlärning kan användas för att sortera användarbeteenden i olika grupper, men om det inte går att fastställa vilken grupp som representerar frustrerat beteende är detta meningslöst.
- Många implementationer av maskininlärning är mycket beräkningstunga jämfört med normalfördelningsmodellen och kräver att data om varje observation sparas. Detta hade medfört svårigheter för mobila enheter med begränsad beräkningskapacitet.

Maskininlärnings algoritmer skulle potentiellt kunna användas för att förbättra precisionen i *EmoTouch* beteendeanalys. En mer ingående undersökning om möjligheterna med maskininlärning för att upptäcka frustration är dock för stor för att rymmas inom arbetets omfång. På grund av en sådan undersöknings storlek i kombination med osäkerheten gällande dess avkastning lämnas detta ämne till förslag för vidareutveckling (avsnitt 8.4.1).

6.2.4 EmoTouch Dashboard

För att kunna visualisera den insamlade datan om applikationsanvändares beteende har webbapplikationen *EmoTouch Dashboard* utvecklats. Med *EmoTouch Dashboard* kan en applikationsutgivare logga in med sitt Google Analytics-konto och ta del av data om sina användares frustrationsnivåer visualiserad med grafer.

EmoTouch Dashboard är en webbapplikation skriven i HTML, CSS och JavaScript. Gränssnittet är i huvudsak uppbyggt av komponenter från Bootstrap 3, vilket har använts för att underlätta arbetet med stilmallar. För att förenkla JavaScript manipulation av HTML-dokumentet används jQuery. För att *EmoTouch Dashboard* ska kunna hämta Google Analytics data autentiserar applikationsutgivaren sig med sitt Google-konto. Därefter hämtas Google Analytics-data via AJAX-anrop med hjälp av Googles JavaScript-bibliotek "Google APIs Client Library for JavaScript”. Data från Google Analytics presenteras i grafer med hjälp av Highcharts.

EmoTouch Dashboard består av följande sidor:

- Inloggningssidan - Innehåller information om *EmoTouch* och funktionalitet för att logga in med ett Google-konto.
- Huvudsidan - Sidan som visas när användaren är inloggad. Innehåller navigering för att byta Google Analytics-konto och Google Analytics-profil.

Huvudsidan innehåller tre flikar:

- Översiktssidan - Visar en översikt för den senaste veckans data.
- Kartsidan - Visar andelen frustrerade användare i olika länder just nu.
- Realtidssidan - Visar antalet aktiva och frustrerade användare just nu.

Här följer en redogörelse av de viktigaste valen som gjordes under implementationsarbetet med *EmoTouch Dashboard*.

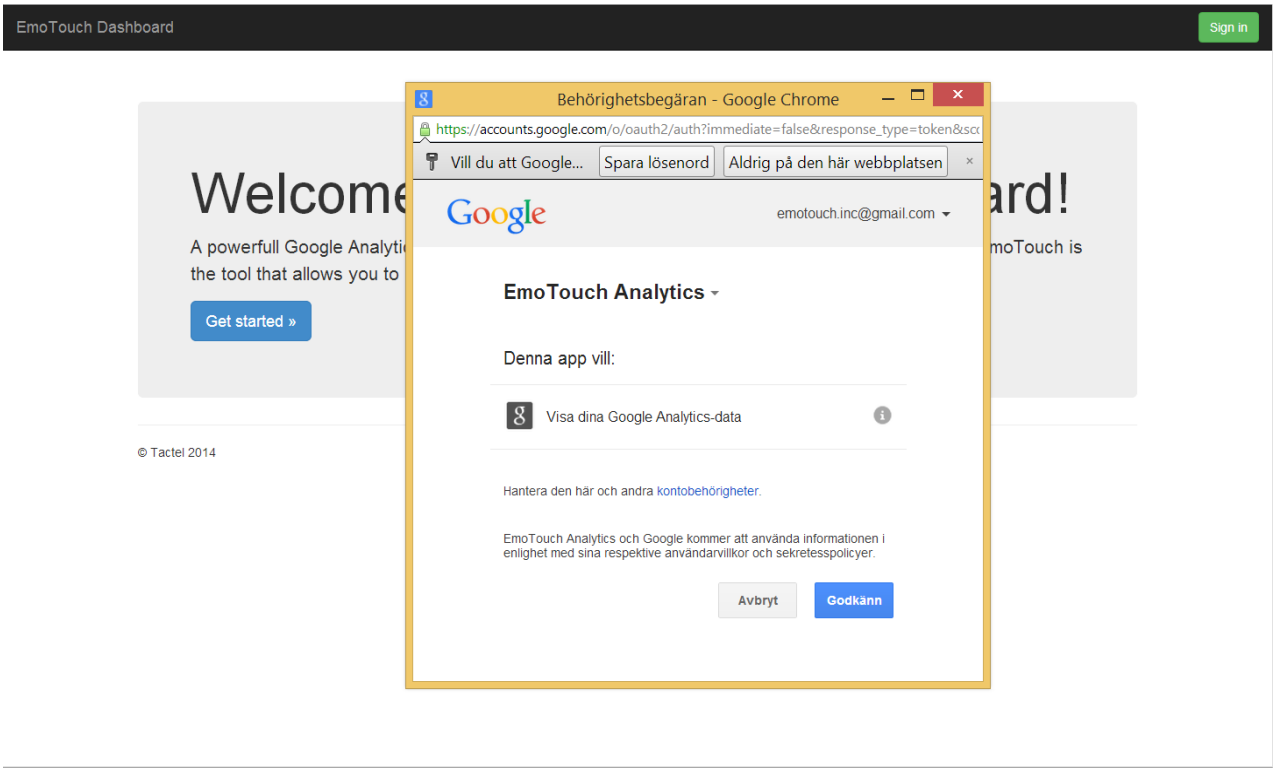
6.2.4.1 Google Analytics befintliga webbapplikation

Google Analytics erbjuder en webbapplikation som tillåter applikationsutgivare att ta fram grafer och tabeller över deras användares interaktion med applikationen. Under utvecklingsfasen av *EmoTouch Android SDK* användes detta för att verifiera att data skickades till Googles server på ett korrekt sätt. Möjligheten fanns att använda Google Analytics webbapplikation som *EmoTouch Dashboard*, men eftersom den inte kunde visualisera den information som var relevant för *EmoTouch* beslutades det att en separat webbapplikation skulle utvecklas från grunden.

6.2.4.2 Inloggning och autentisering

Eftersom *EmoTouch* använder Google Analytics för datahantering måste en applikationsutgivare som vill använda *EmoTouch* ha ett Google-konto och ett konto på Google Analytics, där en eller flera Android-applikationer har registrerats. Detta innebär att användaren kan använda sitt Google-konto för att autentisera sig. För att *EmoTouch Dashboard* sedan ska kunna hämta data från användarens Google Analytics-konto måste den auktoriseras åtkomst av användaren.

All funktionalitet för autentisering och auktorisering tillhandahålls av Google för att utvecklare enkelt ska kunna konstruera den här typen av applikationer. Google använder sig av OAuth 2.0 för användare ska kunna ge tredjeparts applikationer tillåtelse att använda deras data. Figur 13 visar hur *EmoTouch Dashboard* frågar en användare på inloggningssidan om tillåtelse att använda deras Google Analytics data.



Figur 13: EmoTouch Dashboard ber en användare om tillåtelse att använda sig av dennes Google Analytics-data.

6.2.4.3 Hämtning av data från Google Analytics

EmoTouch Dashboard använder Googles Javascript-bibliotek för Core Reporting API (v3) och Real Time Reporting API (v3) för att hämta data asynkront från Google Analytics. Det har observerats att det kan ta mellan 10 och 40 minuter efter att en händelse inträffat innan svaren från Core Reporting API (v3) speglar verkligheten. För Real Time Reporting API (v3) sker det inom en minut i alla observerade fall.

Det finns dock ett problem med att tolka datan i svaren från det sistnämnda. När en användare startar en Android-applikation som har Google Analytics SDK och *EmoTouch Android SDK* integrerad, kommer värdet av datafältet för antalet aktiva användare att öka med ett inom en minuts tid. När användaren sedan stänger av applikationen dröjer det fem minuter innan samma värde minskar med ett. Om användaren däremot utlöser en Google Analytics-händelse, vilket sker när användaren uppvisar ett avvikande beteende, kommer värdet av antalet Google Analytics-händelser just nu öka med ett, och inte minska igen förrän efter ca 8 minuter. API:et erbjuder dessutom ingen möjlighet att filtrera svaren efter tidsintervall. Detta medför att det vid vissa tidpunkter kan se ut att finnas fler frustrerade användare än vad det finns aktiva användare, vilket är problematiskt ur användbarhetssynpunkt.

Detta problem finns inte hos Core Reporting API (v3) eftersom det här går att begära antalet aktiva användare och antalet skickade Google Analytics-händelser summerade för varje minut. Det är heller inget stort problem i ett scenario där *EmoTouch* används för en Android-applikation med tusentals aktiva användare, eftersom antalet användare som uppvisar avvikande beteende troligen kommer vara mycket litet i förhållande till det totala antalet användare. I det här arbetet antas en applikationsutgivare inte främst vara intresserad av exakt hur många procent av ens användare som är frustrerade, utan snarare vilka samband det finns mellan en förändring av frustrationsnivån och övriga faktorer.

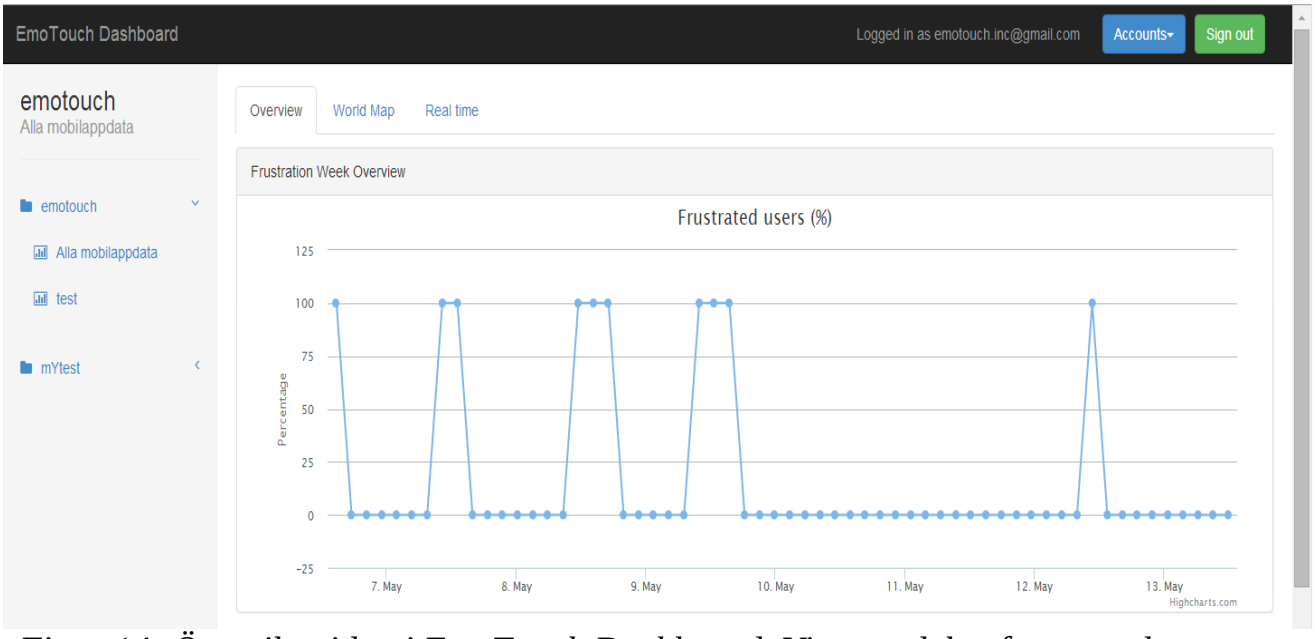
Som tidigare nämnt är Real Time Reporting API (v3) i skrivande stund i beta-stadie. Detta innebär att funktionaliteten som erbjuds kan komma att förändras och utökas i framtiden.

6.2.4.4 Visualisering av data

EmoTouch Dashboard visualiserar statistik genom grafer och diagram som genereras med hjälp av Highcharts. Den information som visas är beskriven nedan.

På *EmoTouch Dashboards* översiktssida finns ett linjediagram som visar andelen frustrerade användare i förhållande till det totala antalet användare, uttryckt i procent, över tid. Med frustrerade användare menas användare hos vilka avvikande beteende har detekterats av *EmoTouch*. Diagrammets tidsintervall är en vecka.

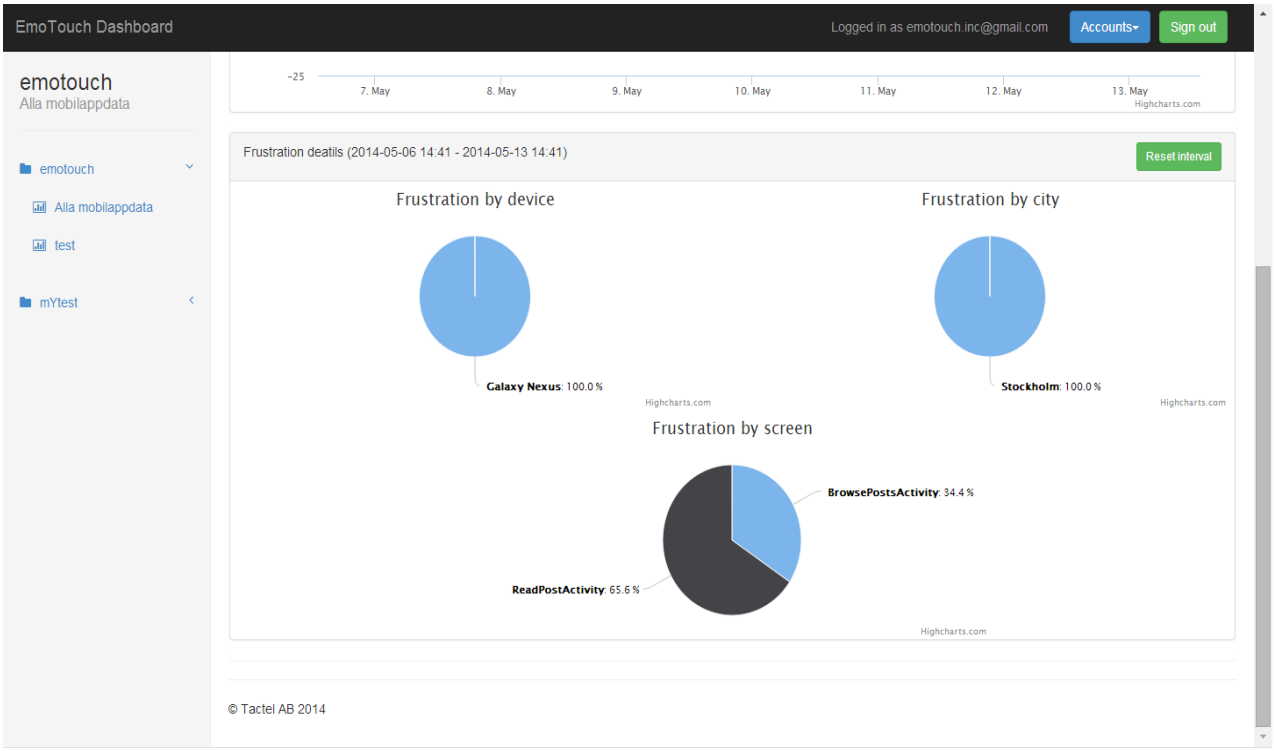
Syftet med detta diagram är att ge applikationsutgivare en översiktsbild av sin användarpopulation. Till exempel om diagrammet visar en stor ökning av andelen frustrerade användare vid ett specifikt tillfälle är det anledning till vidare undersökning. Frustrationsökningen kanske inträffade samtidigt som en ny uppdateringen av applikationen och med denna kunskap kan detta förebyggas i framtiden. Figur 14 visar hur detta diagram ser ut på översiktssidan i *EmoTouch Dashboard*.



Figur 14: Översiktssidan i EmoTouch Dashboard. Visar andelen frustrerade användaren den senaste veckan.

På översiktssidan finns även tre cirkeldiagram som visar frustrerade användares fördelning för viss datatillhörighet. Ett visar fördelning mellan olika modeller av enheter, ett visar fördelning mellan olika städer, och ett visar fördelning mellan vilka av applikationens olika skärmar som var aktiva när frustration detekterades.

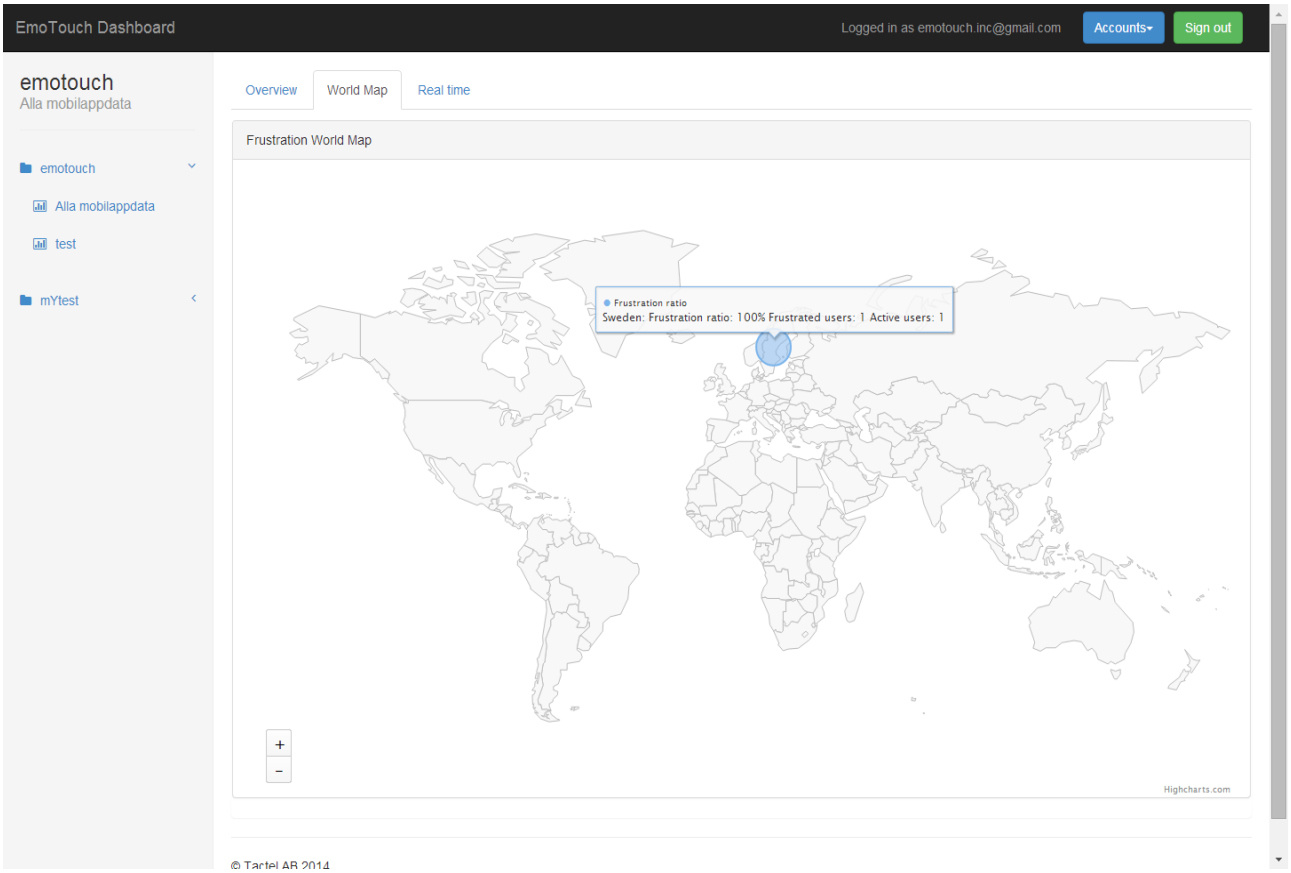
Cirkeldiagrammen ger värdefull information om vilka delar av användarpopulationen som ligger bakom detektionerna av frustration vid ett visst tillfälle. Dessa grafer kan visa vilka de frustrerade användarna är. Till exempel om majoriteten av frustrationsdetektionerna inträffar på en viss skärm i applikationen, så är det en möjlig indikation på användarbarhetsproblem med denna skärm. Figur 15 visar hur de tre diagrammen ser ut på översiktssidan i *EmoTouch Dashboard*.



Figur 15: Översiktssidan i EmoTouch Dashboard. Visar fördelningen av frustrerade användare för användningsenhet, stad och applikationsskärm.

På kartsidan finns en världskarta som visar andelen frustrerade användare i förhållande till det totala antalet användare, uttryckt i procent, för varje land. Information visualiseras i form av en cirkel vars storlek anpassas efter andelen frustrerade användare i landet. Om landet inte har några aktiva användare för tillfället så visas ingen cirkel för det landet på kartan. Informationen på kartan uppdateras var 10:e sekund.

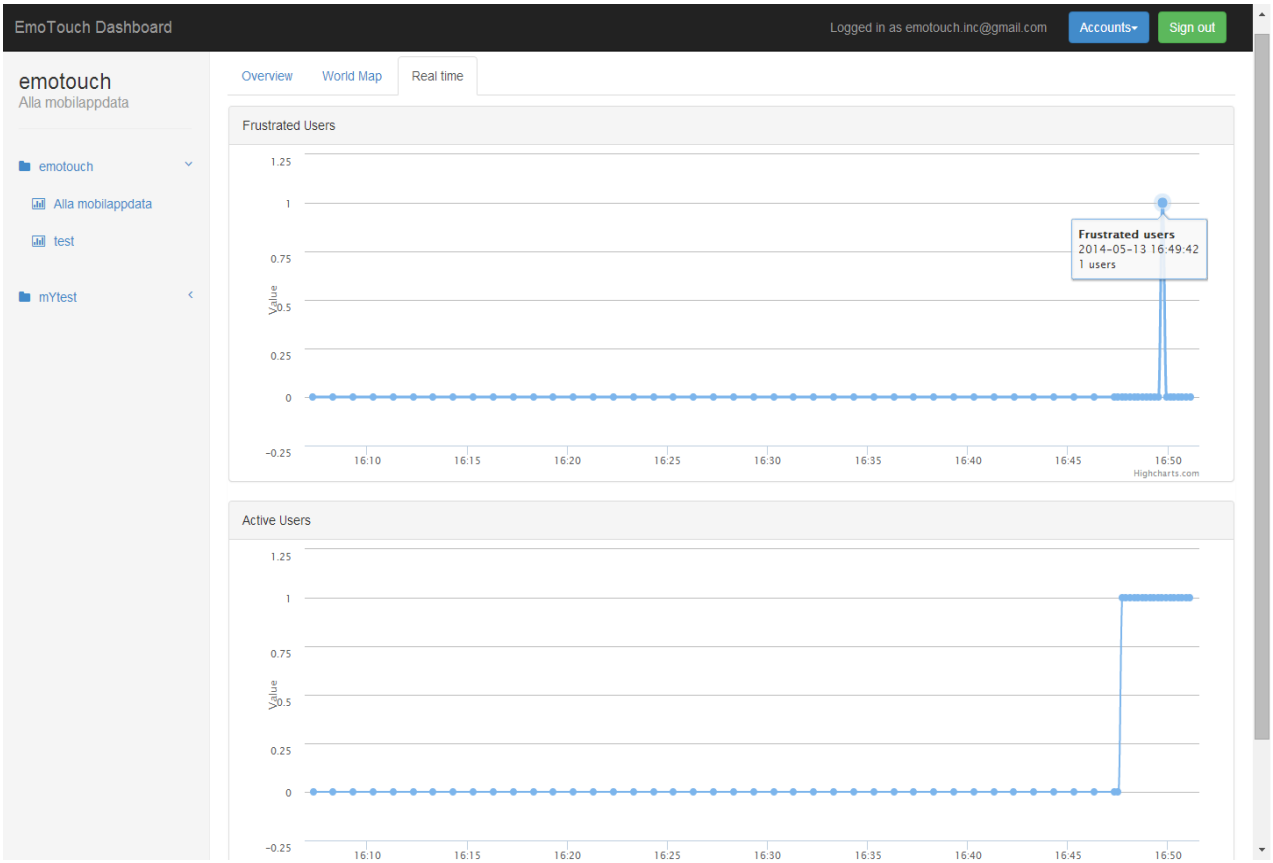
Världskartan kan ge applikationsutgivare som verkar på en internationell nivå en översikt av deras användares frustrationsnivåer just nu. Om andelen frustrerade användare i ett enskilt land plötsligt ökar kraftigt kan detta vara en indikation på lokala problem. Till exempel om problem med en lokal server orsakar svårigheter för användare i ett visst land finns det en möjlighet att upptäcka dessa problem tidigt. Figur 16 visar hur världskartan på kartsidan ser ut i *EmoTouch Dashboard*.



Figur 16: Kartsidan i EmoTouch Dashboard. Visar andelen frustrerade användare just nu i alla länder som har aktiva användare.

På realtidssidan finns två linjediagram som visar antalet aktiva användare, respektive antalet frustrerade användare över tid. De två diagrammen har samma tidsintervall och uppdateras samtidigt. Informationen i graferna uppdateras var 12:e sekund.

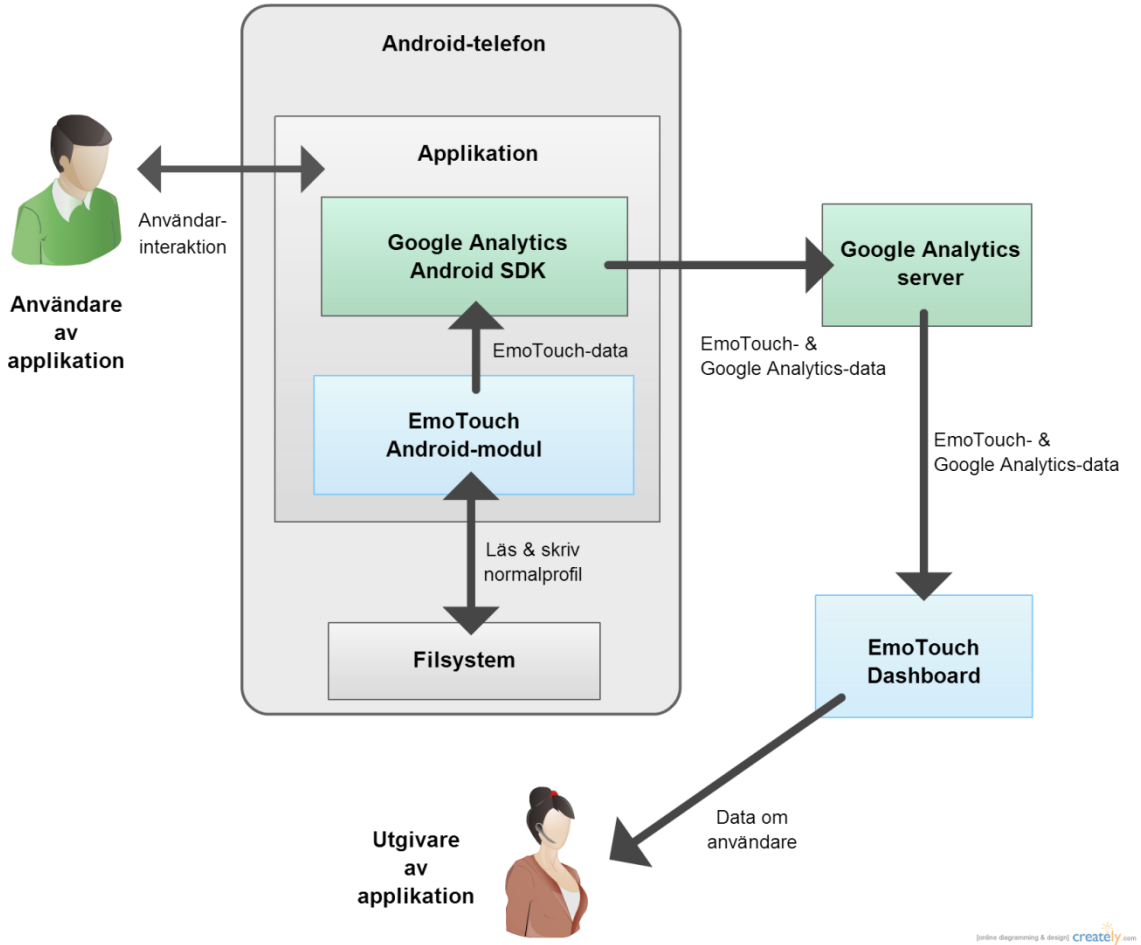
Syftet med linjediagrammen är att ge en bild av hur användarpopulationen beter sig just nu. Genom att observera antalet aktiva användare i förhållande till antalet frustrerade användare kan applikationsutgivare observera och reagera på plötsliga förändringar. Figur 17 visar hur graferna på realtidssidan i *EmoTouch Dashboard* ser ut.



Figur 17: Realtidssidan i EmoTouch Dashboard. Den övre grafen visar antalet frustrerade användare och undre antalet aktiva användare.

6.3 Systemets design

Kontextdiagrammet i figur 18 visar de olika delsystemen som utgör EmoTouch och hur de kommunicerar med varandra och övriga system i sin omgivning.



Figur 18: Kontextdiagram för EmoTouch. EmoTouch utgörs av de ljusblå delarna.

6.3.1 EmoTouch Android SDK

6.3.1.1 UML

Se bilaga A för UML-diagram över *EmoTouch Android SDK*.

6.3.1.2 Paketstruktur och klasser

Se bilaga B för paketstruktur och klasser för *EmoTouch Android SDK*.

6.3.2 EmoTouch Dashboard

6.3.2.1 Beståndsdelar

EmoTouch Dashboards konstruktion har redan beskrivits i detalj under avsnitt 6.2.4. Här följer en överblick av de beståndsdelar som utgör *EmoTouch Dashboard*.

- Ett HTML-dokument där layouten för samtliga sidor i applikationen definieras. Vid navigering mellan sidor skiftas vilka element som är synliga respektive osynliga med hjälp av JavaScript.
- En CSS-fil som definierar utseendet på de grafiska elementen. Utöver denna används CSS-biblioteket Bootstrap för stilmallar.
- Ett antal JavaScript-filer som innehåller kod för manipulering av grafiska element, hantering av förfrågningar till Google Analytics REST API och hantering av auktorisering. Utöver dessa används JavaScript-biblioteken Highcharts, jQuery, Google APIs Client Library for JavaScript och Bootstrap.

6.3.2.2 Flödesdiagram

Se bilaga C för flödesdiagram för *EmoTouch Dashboard*.

7 Test

7.1 Syfte med tester

Under utvecklingsarbetets har det kontinuerligt testats att *EmoTouch* kan detektera statistiska avvikelser i olika beteendeparametrar.
Det är mer komplicerat att testa om *EmoTouch* kan ge någon indikation om en användares nuvarande emotionella reaktioner. Efter att en fungerande *EmoTouch Android SDK* färdigställts utfördes tester med försökspersoner. Försökspersonerna fick interagera med en Android-applikation först på ett normalt sätt, och sedan under ett tillstånd av simulerad frustration.

Syftet var i första hand att undersöka om *EmoTouch* kunde identifiera huruvida användaren var i det normala eller det simulerade frustrerade tillståndet. Utöver detta användes testet för att upptäcka eventuella möjligheter att förbättra systemet för att en högre grad av korrekthet ska kunna uppnås.

7.2 Utförande av tester

Tester utfördes med 6 försökspersoner som fick utföra samma fördefinierade uppgifter i en för testsyfte utvecklad applikation för tjänster inom kollektivtrafik (Se bilaga D “ Testinstruktioner”). Försökspersonerna trodde att de medverkade i ett användartest för att testa användbarheten i applikationen. Under de första fem scenarierna antogs försökspersonen vara i ett icke-frustrerat tillstånd. Inför scenario 6 förklarades testets egentliga syfte: att detektera frustration. Försökspersonen uppmanades att först beskriva sitt typiska användarbeteende i en frustrerad situation, och fick sedan utföra det sista scenariot i ett simulerat frustrerat tillstånd.

7.3 Försökspersoner

Försökspersoner valdes som inte hade någon tidigare kännedom om vad projektet gick ut på. De valdes så att olika användargrupper var representerade med avseende på ålder och kön. Det var också viktigt att försökspersonerna inte studerade eller arbetade inom datateknik, applikationsutveckling eller interaktionsdesign eftersom det då funnits en risk att de gått ur rollen som användare och intagit en mer analytisk inställning till applikationen de använde. Tabell 1 visar information om de olika försökspersonerna.

Försöksperson	Kön	Åldersgrupp	Vana av applikationsanvändning
A	Man	60-70 år	Medel
B	Kvinna	60-70 år	Under medel
C	Man	20-30 år	Medel
D	Kvinna	20-30 år	Medel
E	Kvinna	20-30 år	Medel
F	Kvinna	20-30 år	Under medel

Tabell 1: Kön, åldersgrupp och vana av applikationsanvändning för alla testsubjekt. Vana av applikationsanvändning bygger på självuppskattning.

7.4 Testmiljö och utrustning

EmoTouch Android SDK var integrerad i applikationen och hade modifierats för att generera loggfiler innehållandes utskrifter om alla uppmätta beteendeparametrar och avvikelser från användarens normalprofil. Dessa loggfiler var avsedda att studeras i efterhand för att ge förståelse om de olika beteendeparametrarnas karakteristik. Informationen i loggfilerna skulle visa huruvida *EmoTouch* mätte de olika parametrarna på ett lämpligt sätt, och förhoppningsvis ge någon indikation om hur mätningar kunde förbättras.

Samtliga tester utfördes i hemmiljö. En av testledarna agerade testvärd och den andra agerade observatör. Testvärden förklarade hur testet skulle gå till och gav instruktioner till försökspersonen samt svarade på eventuella frågor. Observatören filmade telefonens skärm och förde anteckningar om försökspersonens beteende.

7.5 Diskussion om testmetoder

Att använda en testapplikation för tjänster inom kollektivtrafik, valdes för att det är ett användningsområde som de flesta användare är bekanta med, och för att det möjliggör ett gränssnitt med många olika typer av användarinteraktion. För att *EmoTouch* ska kunna observera alla möjliga beteendeparametrar bör gränssnittet innehålla tryckbara komponenter som knappar, scrollbara komponenter eller andra komponenter som manipuleras med gester, samt textfält.

En testmetod som diskuterades var att installera en applikation med *EmoTouch Android SDK* integrerad, på försökspersoners egna telefoner och låta dem använda applikationen i sin vardag under en lägre tidsperiod. Därefter skulle de uppmanas att simulera ett frustrerat användarbeteende i applikationen. Det finns två fördelar med denna metod; normalprofilen hade byggts upp under en längre tid och därmed blivit mer tillförlitlig som statistiskt underlag. Dessutom hade det eliminerat risken att försökspersonernas beteende påverkats av testsituationen.

Denna metod avvisades på grund av att det inte gick att hitta en lämplig applikation som uppfyller kraven:

- Applikationens gränssnitt bör innehålla tryckbara komponenter, scrollbara komponenter eller andra komponenter som manipuleras med gester, samt textfält.
- Applikationen bör användas så ofta, och under så lång tid som möjligt. För att antalet observationer av alla parametrar ska bli tillräckligt många måste applikationen användas under en lång tid. Om applikationen används vid många tillfällen ökar sannolikheten att normalprofilen motsvarar ett genomsnitt av användarens beteende vid olika känslotillstånd.
- För att *EmoTouch Android SDK* skulle kunna integreras måste applikationen antingen vara en befintlig applikation med öppen källkod, eller en för ändamålet utvecklad testapplikation som går att utveckla under rimlig tid, och som är tillräckligt funktionell för att en försöksperson ska kunna använda den på ett normalt sätt i sin vardag.

Ett alternativt tillvägagångssätt för utförandet av tester hade varit att försöka provocera fram en äkta känsla av frustration hos försökspersonerna istället för att låta dem simulera ett frustrerat beteende. Detta hade varit problematiskt eftersom det är svårt att förutsäga hur olika personer kommer att reagera känslomässigt på en viss situation. Att låta försökspersonerna simulera sitt typiska frustrerade användarbeteende är dock inte helt utan problematik. En försöksperson kan sakna insikt i hur de brukar bete sig när de är frustrerade, och det kan finnas en risk att de känner sig obekväma med att låtsas vara frustrerade vid testtillfället.

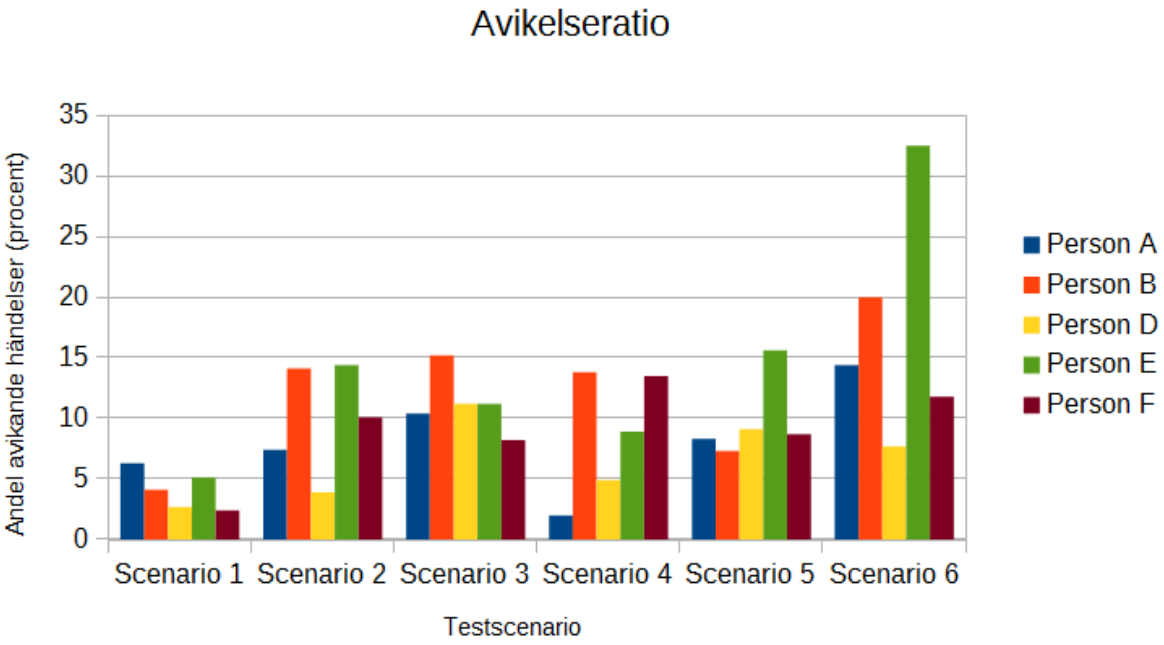
7.6 Resultat av tester

En sammanställning av testresultaten finns att läsa i bilaga E “Testresultat”. Om man studerar antalet tillfällen *EmoTouch* detekterade frustration under varje scenario (se tabell 2) ser det inte ut att finnas något samband mellan användarens simulerade frustrerade tillstånd under scenario 6 och *EmoTouchs* detektioner. På grund av tekniska problem blev resultaten för person C under scenario 6 obrukbara.

Scenario	Person A	Person B	Person C	Person D	Person E	Person F
1	0	0	0	0	0	0
2	0	1	1	0	2	0
3	0	1	0	1	1	0
4	0	0	0	0	0	0
5	1	0	2	0	1	0
6	0	2	?	0	2	0

Tabell 2: Antal detektioner av frustration per scenario för varje testperson.

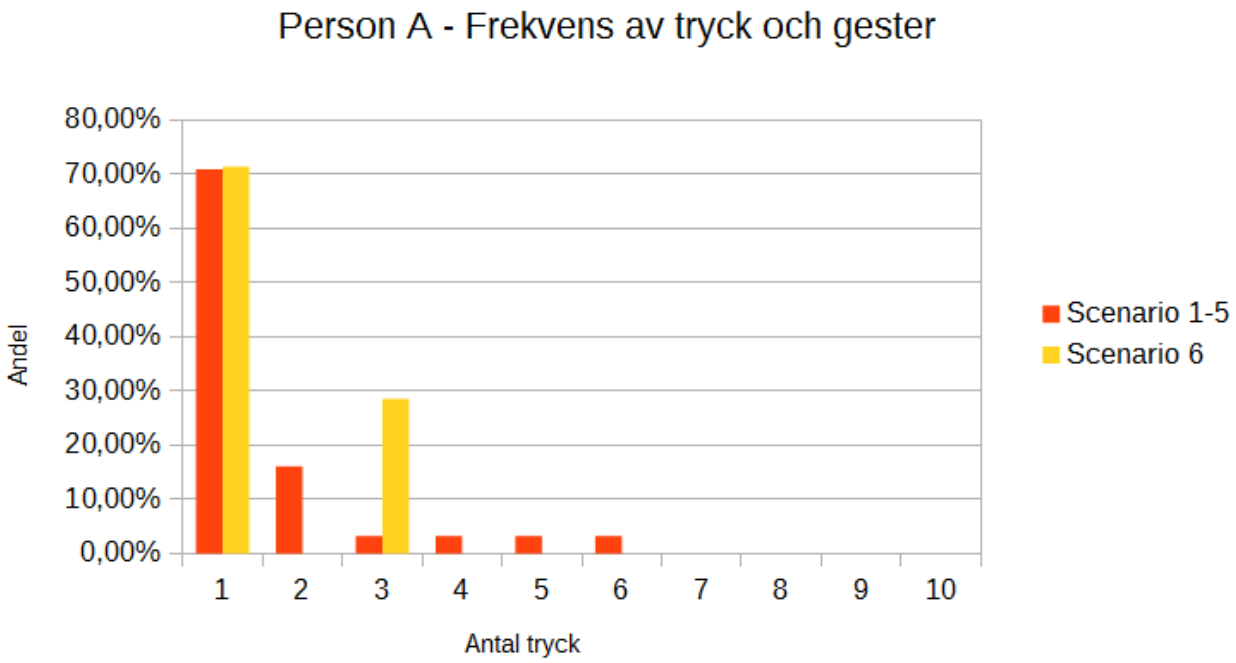
Om man däremot studerar den så kallade avvikelseration, det vill säga andelen avvikande observationer i förhållande till det totala antalet observationer för varje scenario, ser man däremot ett starkt samband. Av fem användare uppnådde tre stycken sitt högsta värde på avvikelseratio under scenario 6. En användare uppnådde sitt näst högsta värde och en annan uppnådde sitt tredje högsta värde. Detta visas i figur 19.



Figur 19. Avvikelseratio för varje försöksperson och scenario.

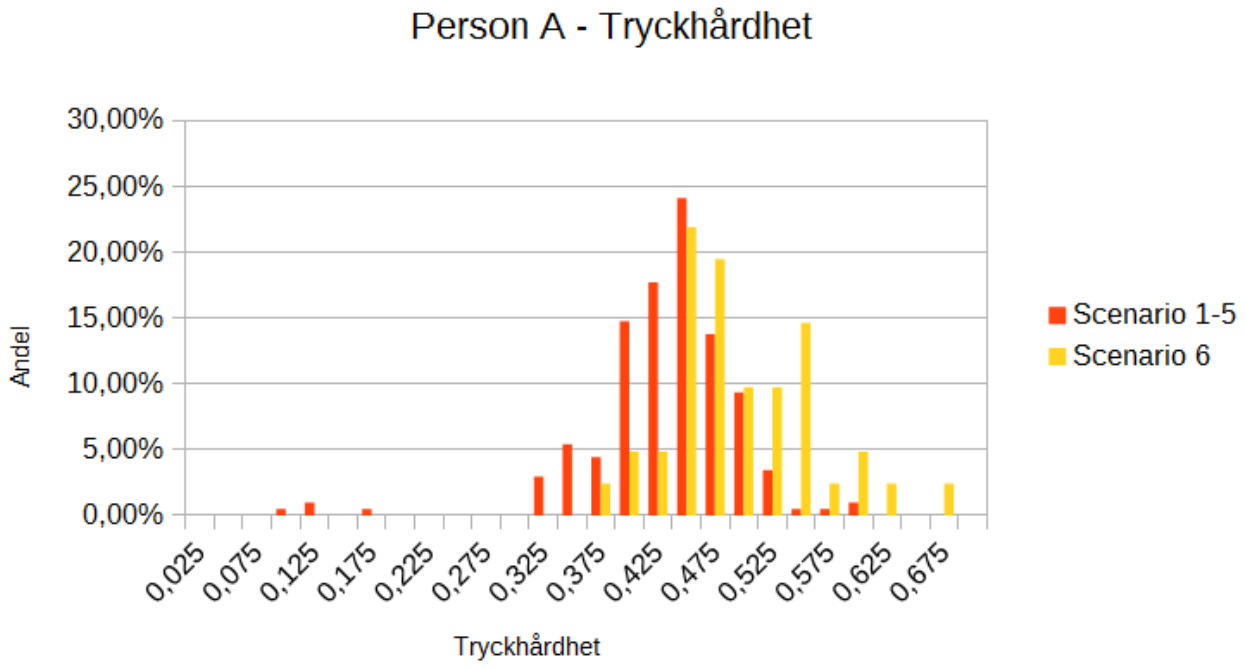
Testerna visade också att tre parametrar är problematiska att beräkna på det sätt som *EmoTouch* beräknar dem.

- För flera användare fick längden av gester ett medelvärde som låg väldigt nära noll och en stor standardavvikelse. Detta medför att det efter en viss tid blev omöjligt för en observation att anta ett värde som är mindre än medelvärdet minus standardavvikelsen, vilket är definitionen för en avvikelse.
- Frekvens av skärmtryck mäts som längden av en sekvens skärmtryck där tiden mellan varje tryck inte får överstiga ett visst statiskt gränsvärde. Längden av sekvensen måste vara minst två skärmtryck för att mätvärdet ska räknas som en observation. Problemet med detta sätt att mäta är att en övervägande majoritet av observationerna blir sekvenser med två knapptryck, och det blir väldigt liten spridning i värdena. Eftersom det är ett diskret värde med liten variation är det olämpligt att använda normalfördelning som statistisk modell. Detta kan man tydligt se i figur 20.
- Raderingsfrekvens vid redigering av textfält mäts på ett sätt liknande frekvens av skärmtryck, och här har samma problem med dålig spridning av värdena observerats.



Figur 20: Ett exempel på en försökspersons fördelning av frekvens av tryck och gester. Observera att värdet 1 på antal tryck betyder en sekvens med 2 skärmtryck.

Om man studerar spridningen av de olika parametrarna (se bilaga D “ Parametervärden”) tycks det som att olika parametrar bär olika signifikans vid detektion av frustration. Den parameter som uppvisar den största skillnaden mellan normalt beteende i scenario 1-5 och frustrerat beteende i scenario 6 är tryckhårdhet. Detta stämmer väl överens med de resultat som presenterades av Mentis (2002) och Gao, Bianchi-Berthouze och Meng (2012). Figur 21 visar hur en försökspersons värden av tryckhårdhet fördelades under experimentet. Förskjutningen åt höger av mätvärden vid scenario 6 visar tydligt att personen tryckte hårdare i ett simulerat frustrerat tillstånd. Observera att tryckhårdhet visade störst signifikans i en version av *EmoTouch* där tre av de övriga parametrarna inte mättes på ett optimalt sätt. I en förbättrad version, där tryckfrekvens, gestlängd och raderingsfrekvens implementerats på ett annorlunda sätt är det möjligt att någon av dessa visar sig ha störst signifikans.



Figur 21: Ett exempel på en försökspersons fördelning av tryckhårdhet.

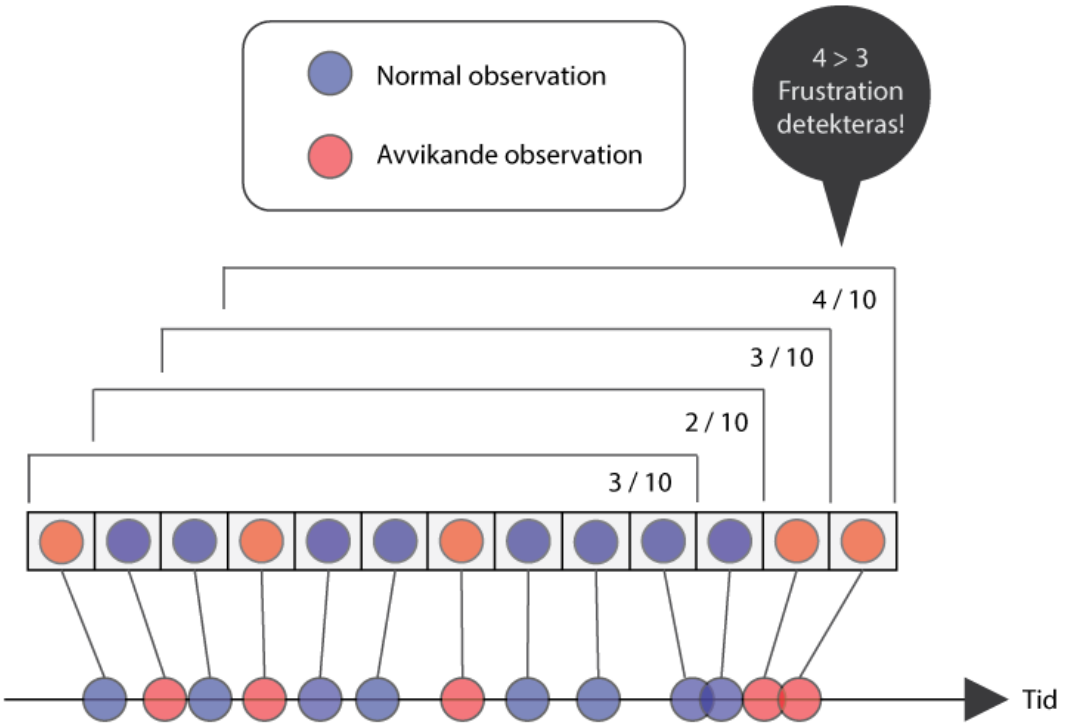
Ett annan intressant resultat av testerna är att det finns en signifikant skillnad mellan de olika försökspersonernas medelvärden och standardavvikelser för samma beteendeparameter. Parametern tryckhårdhet uppvisar en särskilt stor variation, vilket kan förklaras av att två olika mobiltelefoner har använts vid olika tester, och mätning av tryckhårdhet tycks vara implementerat på olika sätt av olika telefontillverkare.

7.7 Slutsatser av tester

Eftersom *EmoTouch* detekterade en högre andel avvikelser under scenario 6 kan man anta att flera av de beteendeparametrar som *EmoTouch* mäter verkligen har en koppling till ett simulerat frustrerat användarbeteende, och att den metod med vilken *EmoTouch* detekterar avvikelser från normalprofilen är effektiv.

Metoden *EmoTouch* använder för att utifrån dessa enskilda avvikande värden detektera frustration har dock visat sig opålitlig. Denna bygger på att avvikande beteende definieras som en sekvens av avvikande observationer där tiden mellan varje observation inte får överstiga ett visst statiskt gränsvärde. Eftersom resultatet av testerna visar att *EmoTouch* inte bara ger många felaktiga detektioner utan också missar att detektera vid det frustrerade användningstillfället, kan problemet troligen inte åtgärdas enbart genom att justera de statiska gränsvärdena. Det behövs antagligen en helt ny definition av avvikande beteende. Ett tänkbart förslag är att bortse från tidsaspekten och utgå från avvikelseration istället: Om X stycken av de Y senaste

observationerna har varit avvikande från normalprofilen har frustration detekterats. X och Y är statiska gränsvärden. Detta illustreras av figur 22.

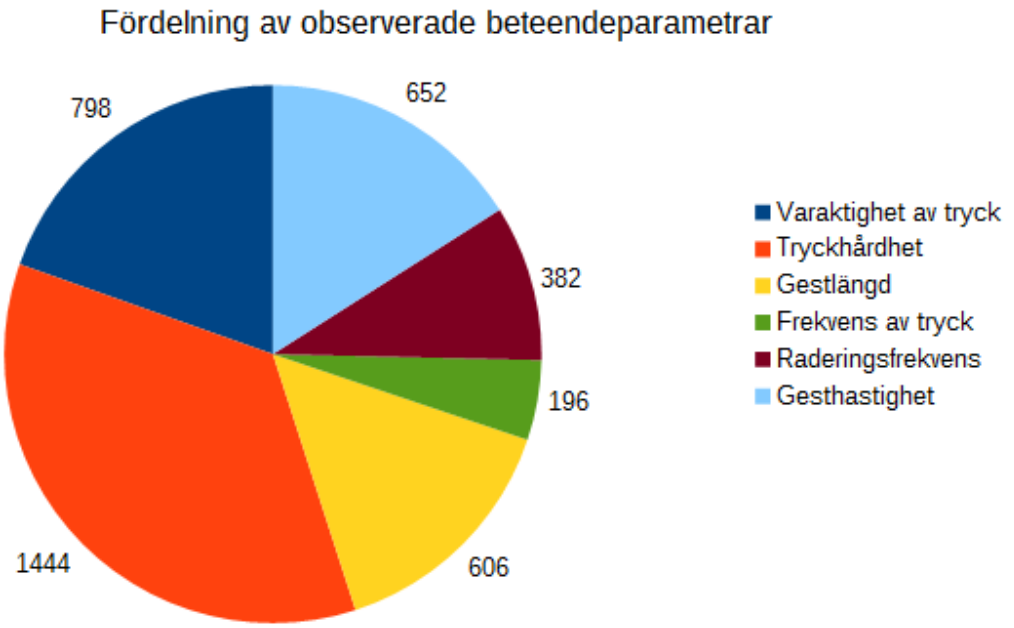


Figur 22: En illustration av en tänkbar alternativ metod att detektera frustration utifrån avvikelseration. I exemplet detekteras frustration när fler än 3 av de senaste 10 observationerna varit avvikande från normalprofilen.

Eftersom olika parametrar inte observeras lika ofta kan det vara lämpligt att titta på avvikelseration för var och en av parametrarna för sig. Annars riskerar de parametrar som observeras oftast att få för stor påverkan på den totala avvikelseration. Man skulle då kunna kombinera avvikelseration för de olika parametrarna exempelvis genom att beräkna ett snittvärde av dem. Eftersom olika parametrar tycks ha olika signifikans för detektering av frustration bör deras avvikelseratio kombineras på ett sådant sätt att de mest signifikanta parametrarna får större påverkan.

Mätningen av frekvens av skärmtryck bör implementeras på ett annorlunda sätt. En möjlighet är att mäta den totala tiden som har passerat mellan det senaste skärmtrycket och skärmtrycket som inträffade X antal skärmtryck tidigare, där X är ett lämpligt statiskt värde. Detta skulle ge ett kontinuerligt värde som är lämpligare att modellera med normalfördelning, än längden av den senaste sekvensen av skärmtryck. Det skulle också ge fler observationer, eftersom det för varje nytt skärmtryck går att räkna ut denna tid. I figur 23 ser man att frekvens av skärmtryck är det värde som har minst antal

observationer. Om den föreslagna ändringen av mätning implementeras borde parametern observeras lika ofta som tryckhårdhet. Fler observationer bör innebära större precision vid utformning av normalprofil och en högre grad av korrekthet i detektion av avvikande beteende.



Figur 23: Fördelningen av antalet observationer för varje parameter summerat för samtliga användare.

Mätningen av längd av gester bör också implementeras på ett annat sätt. Problemet är att de flesta gester är mycket små, med en längd på ett par hundra pixlar, medan vissa gester mäter uppemot 5000 pixlar. Detta inträffade exempelvis när försökspersonen scrollade långsamt fram och tillbaka i en lista med menyval och samtidigt läste om de olika menyvalen. En enkel lösning på problemet hade varit att bortse från gester som överstiger ett visst statiskt gränsvärde, eftersom dessa riskerar att störa normalprofilen.

Eftersom det var en signifikant skillnad på normalprofilen mellan olika försökspersoner får antagandet att en användarspecifik normalprofil ger en högre grad av korrekhet än en användaroberoende, anses bekräftat.

8 Etiska överväganden

Det är inte helt oproblematiskt ur ett etiskt perspektiv att övervaka mobilanvändares beteende för att dra slutsatser om deras känslotillstånd. Många användare hade troligen uppfattat det som integritetskränkande och sett det som en form av spioneri. Även om *EmoTouch* inte gör någon djupgående analys av användarens känsloliv, utan endast letar efter indikationer på frustration, så gör ett sådant system anspråk på en traditionellt uteslutande mänsklig förmåga. Känslor är något som kan uppfattas som mycket privat och att en dator gör förutsägelser om detta kan uppfattas som provocerande. En användare kanske hade velat hålla sin frustration för sig själv, ett system som *EmoTouch* tar bort kontrollen över detta valet från användaren.

Enligt Googles Analytics användarvillkor måste utgivare av webbsidor och applikationer informera sina användare om att de använder Google Analytics (Google Inc. 2014d). Eftersom *EmoTouch* är beroende av Google Analytics för att fungera måste alltså applikationsutgivare som använder sig av *EmoTouch* informera sina användare om att Google Analytics används för att samla in data om dem. Den data *EmoTouch* sparar om en användare skiljer sig något från övrig data som samlas in av Google Analytics. Om denna data står för ett mycket större intrång i användarens privata sfär kan det vara nödvändigt att kräva att utgivare som använder *EmoTouch* i sina applikationer måste informera sina användare om detta, utöver användandet av Google Analytics.

Även om det blir juridiskt tydligt genom att användaren ger applikationen tillåtelse att samla in data så ligger det fortfarande i en etisk gråzon. Det finns ingen garanti att användare är medvetna om exakt vad de går med på när de godkänner datainsamling i en applikation. En undersökning har visat att majoriteten av användare inte läser användaravtal innan de godkänner dem (Sauro, J. 2011). Därmed är det troligt att användarna är omedvetna om att deras känslomässiga reaktioner övervakas även om de har godkänt att *EmoTouch* samlar in data om deras beteende. Vilket lyfter frågan om det är etiskt försvarbart att övervaka användares beteende utan att de är medvetna om det. Man skulle dock kunna hävda att det inte är oetiskt att övervaka en person om personen har givit sitt medgivande till detta. Det är ju trots allt upp till individen att ansvara för vilka applikationer som installeras på ens enhet.

Den etiska problematiken ligger kanske inte främst i vilken data som lagras, utan i de syften för vilka datan används. Med *EmoTouch* har avsikten varit att samla in data för att skapa en bättre användarupplevelse. För detta syftet är det troligt att användarna inte skulle ha några problem med datainsamlingen. Det uppstår dock en etisk problematik med att samla in den här typen av data då

det är svårt att förutsäga vad den kommer användas till i framtiden. Användaravtalet kanske förändras eller datan säljs vidare till andra företag, då är det möjligt att datan används till syften som användarna inte skulle ha godkänt om de hade vart kända från början. Detta är dock ej en problematik som är specifik för *EmoTouch* utan generell för alla former av storskalig datalagring.

9 Diskussion

9.1 Slutsatser

I relation till projektets syfte och målsättning, att undersöka möjligheten att utveckla ett analysverktyg som detekterar simulerad frustration, har tre viktiga resultat uppnåtts:

- En modul som är integrerbar i en godtycklig Android-applikation har konstruerats. Denna modul mäter, kartlägger och analyserar i realtid användarbeteenden relevanta för att dra slutsatser om en användares emotionella reaktioner. Detta visar att det är tekniskt möjligt att skapa ett system som kartlägger en användares normalbeteende och i realtid detekterar avvikande beteende.
- Testresultaten visade ett visst samband mellan andelen avvikelser i beteendeparametrars mätvärden och ett simulerat frustrerat tillstånd. Även om detta samband var litet, så är det en stor framgång i förhållande till uppgiftens komplexitet och projektets omfattning.
- Det har visat sig möjligt att konstruera systemet som ett tillägg till Google Analytics, och att visualisera data från mätmodulerna i en webbapplikation. Detta demonstrerar möjligheten att bygga ett komplett analysverktyg kring den data som modulen samlar in. Genom att utnyttja Google Analytics befintliga plattform och utförliga datainsamling blir *EmoTouch* ett kraftfullare och trovärdigare analysverktyg.

Att dessa resultat har uppnåtts utgör grundförutsättningarna för att det ska vara genomförbart att utveckla ett analysverktyg som detekterar frustration.

De största bristerna i förhållande till projektets syfte och målsättning är följande:

- Sambandet mellan simulerad frustration och detekterade avvikelser vid tester, var stort i förhållande till uppgiftens komplexitet och projektets förutsättningar, men alldeles för litet för att säkert kunna konstatera att det är möjligt att utveckla ett fungerande analysverktyg som detekterar frustration. Sambandet är inget bevis för att *EmoTouch* mäter rätt beteendeparametrar på rätt sätt. Det finns också en stor osäkerhet i hur *EmoTouch* hade klarat av att detektera genuint frustrerat beteende i en verklig användningssituation.
- Det upptäcktes vissa svårigheter med att visualisera data i realtid på grund av begränsningar i Google Analytics REST API, som innebär att det tillkommer en odefinierad fördröjning mellan datainsamling och visualisering. Beroende på användningsområdet för *EmoTouch* kan detta innebära ett problem. Situationen kan dock förändras i framtiden beroende på hur Google Analytics utvecklas.

9.1.1 Kartläggning av användarbeteende

I det här arbetet har en prototyp som kan integreras i en godtycklig Android-applikation och mäta sex olika beteendeparametrar, utvecklats. Mätning sker på ett sätt som utnyttjar Androids inbyggda hantering av användarinmatning, utan att störa applikationens övriga funktionalitet. Kartläggning sker genom att en individuell normalprofil bestående av medelvärde, standardavvikelse och antal observationer för varje parameter sparas till fil och kontinuerligt uppdateras vid nya mätningar. De sex beteendeparametrarna är tryckhårdhet, gesthastighet, varaktighet av skärmtryck, gestlängd, frekvens av skärmtryck och raderingsfrekvens. Tester med försökspersoner visade att de tre förstnämnda parametrarna mäts på ett tillfredsställande sätt, medan de övriga tre uppvisade viss problematik.

Den utvecklade prototypen, *EmoTouch Android SDK*, ger ett svar på den första problemformuleringen, "Hur kan användarbeteenden kartläggas vid användning av en mobilapplikation". *EmoTouch* funktionalitet är dock begränsad till applikationer med viss typ av användarinteraktion. Exempelvis skulle *EmoTouch* inte fungera med en applikation där användaren endast interagerar genom röstkommandon. Detta eftersom *EmoTouch Android SDK* endast kartlägger användarbeteende i form av tryck, gester och textinmatning.

9.1.2 Detektion av avvikande beteende

Vid mätning av beteendeparametrar jämförs mätvärden med den befintliga normalprofilen, och avvikande observationer identifieras i realtid. För att detektera ett användarbeteeende som avviker från det normala kombineras avvikande observationer, och om tillräckligt många inträffat inom tillräckligt kort tid anses ett avvikande användarbeteende ha detekterats.

Tekniken för för att identifiera avvikande användarbeteende ger ett svar på den andra problemformuleringen, "Hur kan dessa kartlagda beteende användas för att detektera avvikande beteende i realitd?". Även om *EmoTouch Android SDK* detekterar avvikande beteende i realtid och genast meddelar Google Analytics om detta så blir inte information om detektioner tillgänglig för applikationsutgivare omedelbart. På grund av tidigare nämnda tekniska begränsningar så blir denna information tillgänglig först efter en viss fördröjning. Detta innebär att detektionen av avvikande beteende inte sker i strikt realtid ur applikationsutgivarens perspektiv.

9.1.3 Detektion av simulerat frustrerat beteende

Resultatet av testerna visade att prototypen inte kan särskilja ett simulerat frustrerat beteende från ett normalt beteende. Det visade sig dock att prototypen detekterade en större andel avvikande observationer vid ett simulerat frustrerat beteende än vid ett normalt beteende, för 3 av 5 försökspersoner. Detta antyder att resultatet av projektet åtminstone är ett steg på vägen mot ett system som kan identifiera simulerad frustration.
Genom att undersöka den tredje problemformuleringen "Hur kan dessa beteendeavvikelser användas för att identifiera ett simulerat frustrerat beteende?" har betydelsefulla upptäckter inom problemområdet gjorts.

Den tredje problemformuleringen är beroende av hur de två föregående har besvarats. Även om *EmoTouch* kartlägger användarbeteenden och detekterar avvikelser, krävs det att rätt beteendeparametrar kartläggs på rätt sätt, och att avvikelser detekteras på rätt sätt, för att det ska finnas en möjlighet att identifiera ett simulerat frustrerat beteende. Urvalet av beteendeparametrar baseras på resultatet av flera studier. Mätningen av beteendeparametrar, och detektionen av avvikande observationer implementerades på så sätt att 3 av 5 försökspersoner uppnådde sin största andel avvikande observationer under det simulerade frustrerade tillståndet. Det är därför troligt att svårigheten att detektera simulerad frustration till stor del orsakats av hur avvikande beteende har identifierats utifrån avvikande observationer.

9.2 Utvärdering av metoder

Angående arbetsprocessen och de metoder som använts är det främst arbetet med tester som borde ha struktureras på ett annorlunda sätt. *EmoTouch Android SDK* konfigurerades för att generera loggfiler där information om alla observationer och avvikelser skrevs ut. Efter utförandet av tester gick väldigt mycket arbetstid åt att sortera ut den relevanta informationen ur loggfilerna och infoga den i kalkylblad för att generera grafer och diagram. Om testerna planerats utförligare skulle mätvärdena skrivas i loggfilen formaterade på ett sådant sätt att de enkelt kunde infogas i kalkylblad. Det hade också varit intressant om loggfilerna innehållit ytterligare information, såsom genomsnittlig tid mellan observationer eller avvikelser.

Vid testning med försökspersoner upptäcktes att tre av sex beteendeparametrar beräknas på ett problematiskt sätt. Detta hade troligen kunnat upptäckas och åtgärdas inför testet om en föregående testomgång utan försökspersoner genomförts, då loggfiler och diagram över mätvärden genererats. Detta hade medfört att testerna med försökspersoner blivit mer givande.

9.3 Möjligheter till vidareutveckling

Under avsnittet Slutsatser av tester (6.4) har vissa mindre förbättringar av mätning av beteendeparametrar samt detektion av avvikande beteende redan föreslagits och diskuterats. Dessa är relativt små förbättringar som går att implementera på ett fåtal arbetsdagar. Det här avsnittet tar upp förslag till vidareutveckling som är mer omfattande.

9.3.1 Förbättrad modell för detektion av frustration

En stor förändring av *EmoTouch Android SDK* hade varit att ersätta eller komplettera den befintliga normalfördelningsmodellen för detektion av avvikande beteende mot en modell som bygger på en maskininlärningsalgoritm. Skillnaderna mellan modellerna har redan diskuterats under avsnittet Modell för detektion av avvikande beteende (5.4). Det som följer är en idé om hur denna förändring hade kunnat genomföras.

Låt en stor grupp försökspersoner utföra experiment liknande det beskrivet av Gao, Bianchi-Berthouze och Meng (2012). Försökspersonerna får använda en mobilapplikation samtidigt som information om deras olika beteendeparametrar sparas. Under tiden mäts deras frustrationsnivå antingen genom självuppskattning eller genom fysiologiska indikationer som puls och andningsfrekvens. Sedan låter man en maskininlärningsalgoritm analysera den insamlade datan för att hitta samband mellan frustrerat tillstånd och beteendeparametrar. För att resultatet ska bli oberoende av den enskilda användarens beteendemönster bör analysen ske med avseende på observerade parametrars avvikelse i förhållande till deras normala värde för användaren. Istället för att koppla vissa värden av tryckhårdhet till frustration bör alltså frustration kopplas till avvikelser från användarens normala tryckhårdhet.

EmoTouch skulle då kunna bygga användarens normalprofil och detektera observationer som avviker från denna på det befintliga sättet, men använda maskininlärningsalgoritmen för att identifiera frustration utifrån de avvikande observationerna.

Utöver att analysera endimensionella beteendeparametrar hade *EmoTouch* kunnat analysera mer komplexa beteenden såsom sekvenser av handlingar. För att implementera detta på ett meningsfullt sätt måste först utförliga tester med försökspersoner, som de beskrivna ovan, utföras. Om det går att finna användaroberoende samband mellan frustration och vissa sekvenser av handlingar, hade *EmoTouch* kunnat utnyttja detta för att detektera frustration.

9.3.2 Övrig vidareutveckling

9.3.2.1 Visualisering av händelseförlopp

Den befintliga versionen av *EmoTouch Android SDK* tillåter att utvecklaren definierar *EmoTouch*-händelser som utlöses vid speciella tillfällen i applikationen. Tanken med detta var att underlätta för applikationsutgivaren genom att skapa förståelse för sammanhanget då frustration detekterats. Det kan vara intressant att veta om frustration ofta föregås av en viss sekvens av händelser, eller om frustration ofta leder till en viss sekvens av händelser.

Den befintliga versionen av *EmoTouch Dashboard* erbjuder ingen information om vilka *EmoTouch*-händelser som har utlösts under ett frustrerat användningstillfälle. En möjlig vidareutveckling hade varit att visualisera en valfri användares sekvens av handlingar fram till och efter att frustration har detekterats. Det hade också varit intressant att visualisera statistik över sekvenser av handlingar för stora användargrupper. Hur många procent av de frustrerade användarna har utfört en viss sekvens av handlingar innan frustration detekterades? Hur många procent utförde en viss sekvens efter att frustration detekterades?

9.3.2.2 Varningsmeddelanden

Om *EmoTouch* ska användas för att identifiera frustration i realtid för att lämpliga åtgärder snabbt ska kunna vidtas, kan det vara meningsfullt om applikationsutgivaren kan meddelas exempelvis när frustrationsnivån överstiger ett visst tröskelvärde. Detta skulle kunna göras genom att ett mail eller SMS automatiskt skickas till ansvarig person. För att detta ska kunna implementeras krävs ett program som kontinuerligt hämtar data från Google Analytics och skickar meddelanden när det intressanta värdet överstiger gränsvärdet.

10 Terminologi

10.1 Allmän terminologi

Term	Betydelse
SDK	Förkortning för Software Development Kit. En uppsättning utvecklingsverktyg som möjliggör för mjukvaruutvecklare att bygga applikationer mot en specifik teknisk plattform
REST API	Förkortning för Representational State Transfer Application Programming Interface. Ett REST API definierar ett gränssnitt för utbyte av data mellan en HTTP-server och andra applikationer
AJAX	Förkortning för Asynchronous JavaScript And XML. Ett samlingsnamn för olika tekniker som möjliggör för webbapplikationer att skicka HTTP-förfrågningar asynkront. Möjliggör att script på klientsidan kan kommunicera med servern utan att sidan laddas om.

10.2 Rapportspecifik terminologi

Term	Betydelse
EmoTouch	Namnet på systemet som utvecklas i det här projektet. Omfattar en Android-modul och en webbapplikation
EmoTouch Dashboard	Namnet på webbapplikationen som utvecklas i det här projektet
EmoTouch Android SDK	Namnet på Android-modulen som utvecklas i det här projektet
EmoTouch-händelse	Benämning för en egendefinierad händelse i *EmoTouch Android SDK*. Utvecklare kan lägga till dessa på valfria ställen i källkoden till sin applikation, och skicka valfri data till Google Analytics server
Google Analytics-	Benämning för en egendefinierad händelse i Google

händelse	Analytics Android SDK. Utvecklare kan lägga till dessa på valfria ställen i källkoden till sin applikation, och skicka valfri data till Google Analytics server
Vy	Benämning för komponenten *View* i Android som representerar olika grafiska element i användargränssnittet
Vygrupp	Benämning för komponenten *ViewGroup* i Android som representerar grupper av grafiska element i användargränssnittet
Aktivitet	Benämning för komponenten *Activity* i Android som representerar en skärm i applikationen

11 Referenser

Anthony, L., Carrington, P., Chu, P., Kidd, C., Lai, J. & Sears, A. (2011). Gesture Dynamics: Features Sensitive to Task Difficulty and Correlated with Physiological Sensors. DOI: 10.1145/2370216.2370265

Appsee (2014a). Appsee Mobile Analytics. http://www.appsee.com [2014-02-24]

Appsee (2014b). User Recordings | Appsee. http://www.appsee.com/features/ [2014-02-24]

Banks, T. (2012). MAC and IP Addresses: Personal Information? Privacy and Data Security Law [blogg], 24 juli. http://www.privacyanddatasecuritylaw.com/mac-and-ip-addresses-personal-information [2014-04-08]

Countly (2014). Countly Mobile Analytics. http://count.ly/products/features/ [2014-02-24]

Gao, Y., Bianchi-Berthouze, N. & Meng, H. (2012). What does touch tell us about emotions in touchscreen-based gameplay?. DOI: 10.1145/2395131.2395138

Google Inc. (2013). Android Native Application Tracking Overview. http://developers.google.com/analytics/devguides/collection/android/ [2014-02-24]

Google Inc. (2014a). Mobile App Behavior. http://support.google.com/analytics/answer/2568878?hl=en&ref_topic=2568641 [2014-02-24]

Google Inc. (2014b). MotionEvent | Android Developers. http://developer.android.com/reference/android/view/MotionEvent.html [2014-02-26]

Google Inc. (2014c). Dimensions & Metrics Reference. http://developers.google.com/analytics/devguides/reporting/core/dimsmets [2014-04-08]

Google Inc. (2014d). Google Analytics Terms of Service. http://www.google.com/analytics/terms/us.html [2014-04-08]

Google Inc. (2014e). UUID | Android Developers. http://developer.android.com/reference/java/util/UUID.html [2014-04-08]

Google Inc. (2014f). What Is The Core Reporting API - Overview. http://developers.google.com/analytics/devguides/reporting/core/v3 [2014-05-12]

Google Inc. (2014g). What Is The Real Time Reporting API - Overview. http://developers.google.com/analytics/devguides/reporting/realtime/v3 [2014-05-12]

Google Inc. (2014h). Using OAuth 2.0 to Access Google APIs. https://developers.google.com/accounts/docs/OAuth2 [2014-05-12]

Google Inc. (u.å.a). Android, the world's most popular mobile platform | Android Developers. http://developer.android.com/about/index.html [2014-02-26]

Google Inc. (u.å.b). Application Fundamentals | Android Developers. http://developer.android.com/guide/components/fundamentals.html [2014-02-26]

Google Inc. (u.å.c). UI Overview | Android Developers. http://developer.android.com/guide/topics/ui/overview.html [2014-02-26]

Google Inc. (u.å.d). Input Events | Android Developers. http://developer.android.com/guide/topics/ui/ui-events.html [2014-02-26]

Google Inc. (u.å.e). Processes and Threads | Android Developers. http://developer.android.com/guide/components/processes-and-threads.html [2014-02-26]

Hardt, D. (2012) The OAuth 2.0 Authorization Framework, RFC 6749. http://tools.ietf.org/html/rfc6749 [2014-05-12]

Highsoft AS. (2014) Highcharts - Interactive JavaScript charts for your webpage. http://http://www.highcharts.com/ [2014-05-12]

The jQuery Foundation. (2014). jQuery https://http://jquery.com/ [2014-05-12]

Mentis, H.M. & Gay, G.K. (2002). Using TouchPad Pressure to Detect Negative Affect. DOI: 10.1109/ICMI.2002.1167029

Mitchell, T.M (1997). Machine Learning. Okänd ort: McGraw-Hill Science/Engineering/Math.

Otto, M. & Thornton, J. (2014). Bootstrap http://getbootstrap.com/ [2014-05-12]

Sauro, J. (2011). Do Users Read License Agreements? Measuring Usability - Quantitative Usability, Statistics & Six Sigma by Jeff Sauro [blogg], 11 Januari. http://www.measuringusability.com/blog/eula.php [2014-05-15]

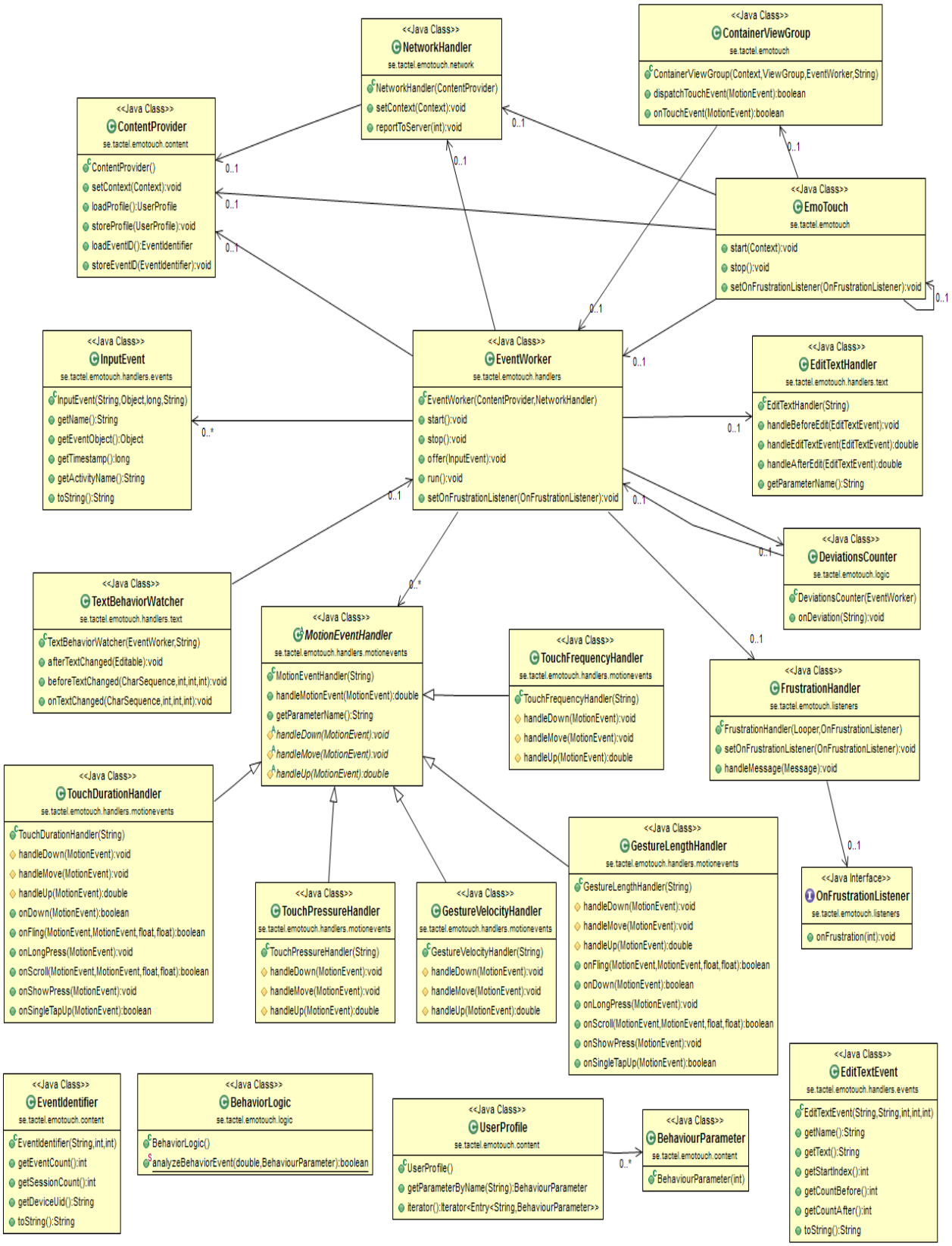
Scheirer, J., Fernandez, R., Klein, J. & Picard, R.W. (2001) Frustrating the user on purpose: a step torward building an affective computer. DOI: 10.1016/S0953-5438(01)00059-5

Smith, D. (2013). Mastering the Android Touch System. http://devsbuild.it/files/PRE_andevcon_mastering-the-android-touch-system.pdf [2014-02-26]

Tsihrintzis, G. A., Virvou, M., Alepis, E. & Stathopoulou, I. (2008) Towards Improving Visual-Facial Emotion Recognition through Use of Complementary Keyboard-Stroke Pattern Information. DOI: 10.1109/ITNG.2008.152

Bilagor

Bilaga A: UML för EmoTouch Android SDK



Bilaga B: Paket och klasser för *EmoTouch Android SDK*

Här följer en beskrivning av de paket och klasser som *EmoTouch Android SDK* består av.

Paket se.tactel.emotouch
Huvudpaketet för *EmoTouch Android SDK*, innehåller alla andra paket och klasser centrala för modulens funktion. Innehåller följande klasser:

- **Klass ApplicationConstants.java**
  Innehåller konstanter för konfiguration av modulen, konstanterna i klassen påverkar hur beteendevariabler mäts och hur avvikelser detekteras.

- **Klass EmoTouch.java**
  En singleton-klass som är gränssnittet till resten av systemet. Anropas för att starta och stoppa *EmoTouchs* beteendeövervakning.

- **Klass EmoTouchViewGroup.java**
  Används för att fånga upp *MotionEvents* i en applikation genom att läggas till som ny rot i applikationens *Vy*-hierarki. Ärver Android-klassen *ViewGroup*.

Paket se.tactel.emotouch.content
Innehåller klasser som hanterar lagring och hämtning av användarens normalprofil. Innehåller följande klasser:

- **Klass ContentProvider.java**
  Används för att hantera läsning och skrivning till fil, för hanteringen av användarens normalprofil.

- **Klass EventIdentifier.java**
  Innehåller information som används för att markera Google Analytics-händelser med ett unikt id för session och händelse.

- **Klass UserProfile.java**
  Representerar användarens sparade normalprofil. Innehåller en lista med de olika beteendeparametrarnas medelvärde, standardavvikelse och antal observationer. Används för att inte behöva läsa och skriva till fil varje gång normalprofilen ska uppdateras.

Paket se.tactel.emotouch.handlers
Innehåller klasser för *EmoTouchs* hantering av olika typer av händelser orsakade av användarinmatning i applikationen. Innehåller följande klass:

- **Klass EventWorker.java**
  EmoTouchs arbetstråd, tar emot händelser av typen InputEvent från Androids UI-tråd via en *mailbox* och hanterar de i den ordningen de kommer. Tråden startas när *EmoTouch* aktiveras och stoppas när användaren lämnar applikationen. Tråden anropar respektive *Handler* beroende på typ av händelse. Har även ansvaret att göra anrop för att ladda och uppdatera normalprofilen och anropa nätverkshanteraren om frustration inträffar.

Paket se.tactel.emotouch.handlers.events
Innehåller klasser som används för *mailbox*-kommunikation mellan UI-tråden och EventWorker-tråden. Innehåller följande klasser:

- **Klass InputEvent.java**
  Används för att skicka ett meddelande till EventWorker-tråden, innhåller information om en händelse och ett objekt som representerar själva händelsen. Objektet som representerar händelsen varierar beroende på typ av händelse.

- **Klass EditTextEvent.java**
  Representerar en händelse som skapas då en användare editerar text i applikationen. Innehåller information om hur texten har modifierats.

Paket se.tactel.emotouch.handlers.motionevents
Innehåller klasser som hanterar olika apsekter av skärmhändelser, så kallade *MotionEvents*. Innehåller följande klasser:

- **Klass MotionEventHandler.java**
  Abstrakt klass för generell hantering av *MotionEvents*.

- **Klass GestureLengthHandler.java**
  Hanterar mätning av en skärmgests längd, beräknar den sammanlagda längden för gesten genom att hantera alla dess *MotionEvents*. Ärver MotionEventHandler.

- **Klass GestureVelocityHandler.java**

Hanterar mätning av en skärmgests hastighet, beräknar medelhastigheten för alla *MotionEvents* i gesten. Ärver MotionEventHandler.

- **Klass TouchDurationHandler.java**
  Hanterar mätning av hur lång tid ett skärmtryck pågår, beräknar skillnaden i tid mellan de *MotionEvents* som startar och avslutar trycket. Ärver MotionEventHandler.

- **Klass TouchFrequencyHandler.java**
  Hanterar mätning av sekvenser av gester, beräknar antalet sammanhängande skärmgester genom att undersöka tiden mellan enskilda gester. Huruvida gesterna anses sammanhängande beror på ett statiskt gränsvärde. Ärver MotionEventHandler.

- **Klass TouchPressureHandler**
  Hanterar mätning av skärmgesters tryckhårdhet, beräknar hårdheten genom att beräkna det högsta värdet av tryckhårdhet bland alla *MotionEvents* för gesten. Ärver MotionEventHandler.

Paket se.tactel.emotouch.handlers.text
Innehåller klasser som hanterar händelser orsakade av textinmatning i applikationen. Innehåller följande klasser:

- **Klass EditTextHandler.java**
  Hanterar mätning av raderingsfrekvensen av text i applikationen, beräknar frekvensen genom att mäta antalet textraderingar i förhållande till det totala antalet textförändringar.

- **Klass TextBeahviorWatcher.java**
  En klass som kopplas till alla textfält i applikationen, innehåller metoder som körs varje gång texten i textfält förändras. Implementerar Android-Interfacet *TextWatcher*.

Paket se.tactel.emotouch.listeners
Innehåller klasser för lyssnare som kan kopplas till *EmoTouch*. Dessa lyssnare kan anropas vid olika händelser i modulen. Innehåller följande klasser:

- **Klass FrustrationHandler.java**
  En klass som ser till att lyssnarna anropas från UI-tråden så att det är säkert att exekvera kod som förändrar applikationens gränssnitt. Denna klass tar emot meddelanden från EventWorker-tråden när en händelse

som har en lyssnare kopplat till sig inträffar. Ärver Android-klassen *Handler*.

- **Klass OnFrustrationListener.java**
  Ett Interface för *EmoTouch* frustrationslyssnare, om ett objekt av denna typ kopplas till *EmoTouch* kommer dess metod att anropas varje gång frustration inträffar.

Paket se.tactel.emotouch.logic
Innehåller klasser som hanterar logik för analys av användarens beteende.
Innehåller följande klasser:

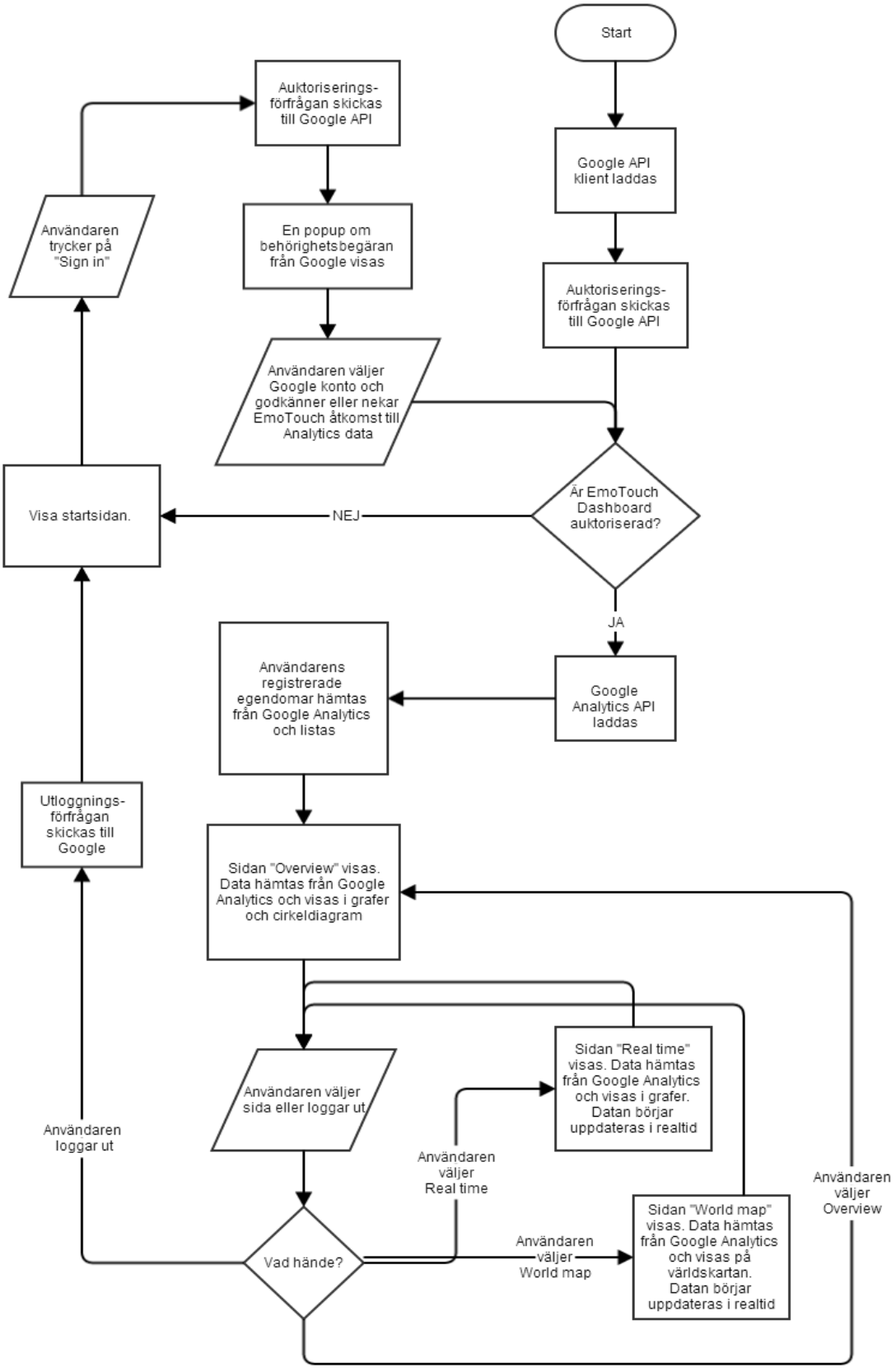
- **Klass BehaviorLogic.java**
  Innehåller statiska metoder för att uppdatera mätvariablernas normalprofiler och detektera eventuella avvikelser bland nya mätningar.

- **Klass DevationsCounter.java**
  En klass som används för att koppla ihop olika detektioner av parameteravvikelser. Genom att undersöka frekvensen och typen av avvikelser görs en bedömning om det nuvarande användarbeteendet kan vara en indikation på frustration.

Paket se.tactel.emotouch.network
Innehåller klasser för att hantera modulens nätverkskommunikation.
Innehåller följande klass:

- **Klass NetworkHandler.java**
  En klass som används för att skicka *Google Analytics*-händelser till Google Analytics server. Används för att skicka händelser när *EmoTouch* upptäcker ett användarbeteende som indikerar frustration, och för att skicka *EmoTouch-händelser*.

Bilaga C: Flödesschema för *EmoTouch Dashboard*



Bilaga D: Testinstruktioner

Testinstruktioner

Scenario 1

Du befinner dig i Tomelilla och vill ta dig till Copenhagen Airport med buss eller tåg.

Uppgift: Använd appen för att hitta en resa och köpa en biljett. Försök att hitta den billigaste resan du kan.

Scenario 2

Tisdagen den 2 april 2014 skulle du ta tåget från Hässleholm C till Malmö C. Tåget skulle avgå kl 18.12 enligt tidtabellen men avgick inte förrän kl 19.15. Enligt trafikbolagets resegaranti har du rätt till ersättning om du gör en anmälan om en försenad resa.

Uppgift: Använd appen för att anmäla förseningen.

Scenario 3

Du befinner dig i Svedala och vill ta dig till Lund C med buss eller tåg. Dagens datum är 16 februari 2014 och du vill vara framme i Lund senast kl 17.30.

Uppgift: Använd appen för att hitta en resa och köpa en biljett. Försök att hitta den senaste resan som tar dig till Lund C senast kl 17.30.

Scenario 4

Du bor och arbetar i Malmö och brukar ta bussen till och från jobbet varje dag. Du har ett resekort som låter dig resa fritt inom Malmö och som laddas periodvis. Du är på väg till bussen en morgon och kommer på att ditt resekort blev ogiltigt igår och måste laddas på med en ny period.

Uppgift: Använd appen för att ladda ditt resekort med en ny period.

Scenario 5

Du bor i Höör och ska börja läsa en kurs i Svalöv på söndagar kl 14.00. Du vill veta vilken tid bör åka hemifrån om du vill ha en resväg med så få byten som möjligt.

Uppgift: Använd appen för att hitta resan med minst antal byten mellan Höör och Svalöv, som anländer före kl 14.00 på en söndag (Tidtabellen ser likadan ut alla söndagar). Du behöver inte genomföra ett biljettköp.

Kort diskussion om testet
innan sista scenariot...

Scenario 6

Du befinner dig vid Lomma busstation och vill ta nästa buss till Malmö Konserthuset för att hinna i tid till ett viktigt möte. Du har sprungit för att hinna med bussen, men upptäcker när du ska gå på att du inte hittar ditt resekort. Busschauffören är mycket otrevlig och låter dig inte gå på bussen förrän du kan uppvisa en giltig biljett.

Uppgift: Använd appen för att köpa en biljett för resan. Välj den resa som ligger närmst i tiden.

Din information

Inloggningsuppgifter

Användarnamn	resenär
Lösenord	password123

Personuppgifter

Använd ditt eget namn, adress etc. Kom ihåg att appen inte sparar några av dessa uppgifter.

Kontokort

Korttyp	Master Card
Kortnummer	4913 5467 8899 0310
CVC-kod	368
Giltigt till	2015-08

Resekort

Kortnummer: 4789 8322 199

Bilaga E: Testresultat

Testresultat

Det som följer är en sammanfattning av resultaten av testerna som genomfördes på försökspersoner. I tabellerna visas de mätvärden som har loggats av EmoTouch Android SDK.

Mätvärde	**Definition**
Observationer	Antalet händelser som givit upphov till en mätbar beteendevariabel. Exempelvis ett tryck på skärmen
Avvikelser	Antalet observationer där en beteendevariabel antagit ett värde som avviker från användarens normalprofil
Avvikelseratio	Antal avvikelser dividerat med antal observationer uttryck i procent
Frustration	Antalet gånger EmoTouch har detekterat frustration

Försöksperson A

Scenario	Observationer	Avvikelser	Avvikelseratio	Frustration
1	173	11	6,3	0
2	216	16	7,4	0
3	144	15	10,4	0
4	49	1	2,0	0
5	156	13	8,3	1
6	134	19	14,4	0

Försöksperson B

Scenario	Observationer	Avvikelser	Avvikelseratio	Frustration
1	146	6	4,1	0
2	192	27	14,1	1
3	125	19	15,2	1
4	58	8	13,8	0
5	110	8	7,3	0
6	120	24	20,0	2

Försöksperson C

På grund av oförutsedda tekniska problem saknas loggfilen för scenario 6.

Scenario	Observationer	Avvikelser	Avvikelseratio	Frustration
1	114	11	9,6	0
2	110	21	19,1	1
3	93	23	24,7	0
4	41	13	31,7	0
5	99	22	22,2	2
6	?	?	?	?

Försöksperson D

Scenario	Observationer	Avvikelser	Avvikelseratio	Frustration
1	146	4	2,7	0
2	127	5	3,9	0
3	125	14	11,2	1
4	61	3	4,9	0
5	110	10	9,1	0
6	78	6	7,7	0

Försöksperson E

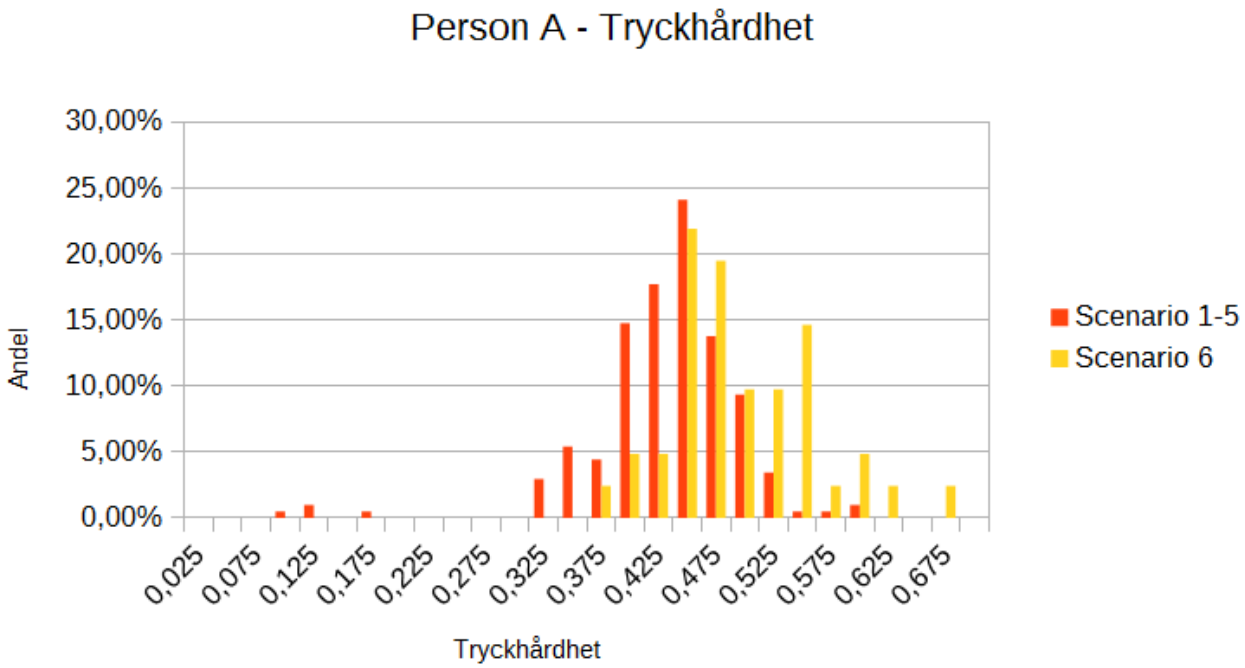
Scenario	Observationer	Avvikelser	Avvikelseratio	Frustration
1	98	5	5,1	0
2	128	27	14,4	2
3	143	16	11,2	1
4	45	4	8,9	0
5	90	14	15,6	1
6	114	37	32,5	2

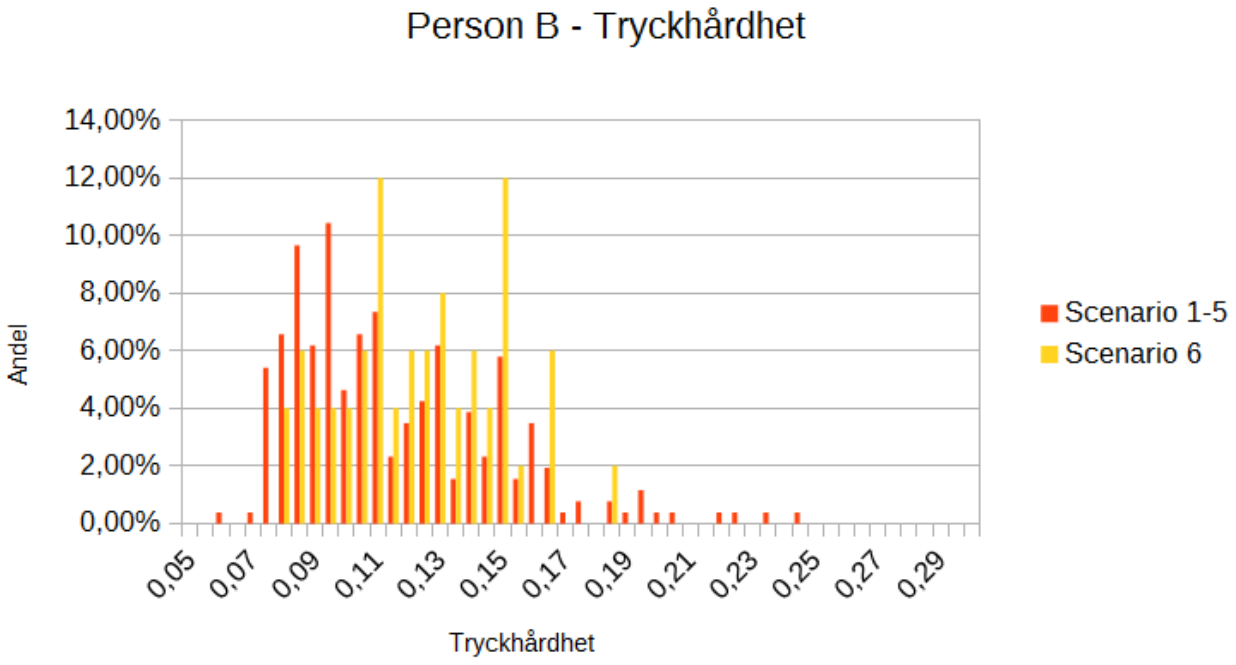
Försöksperson F

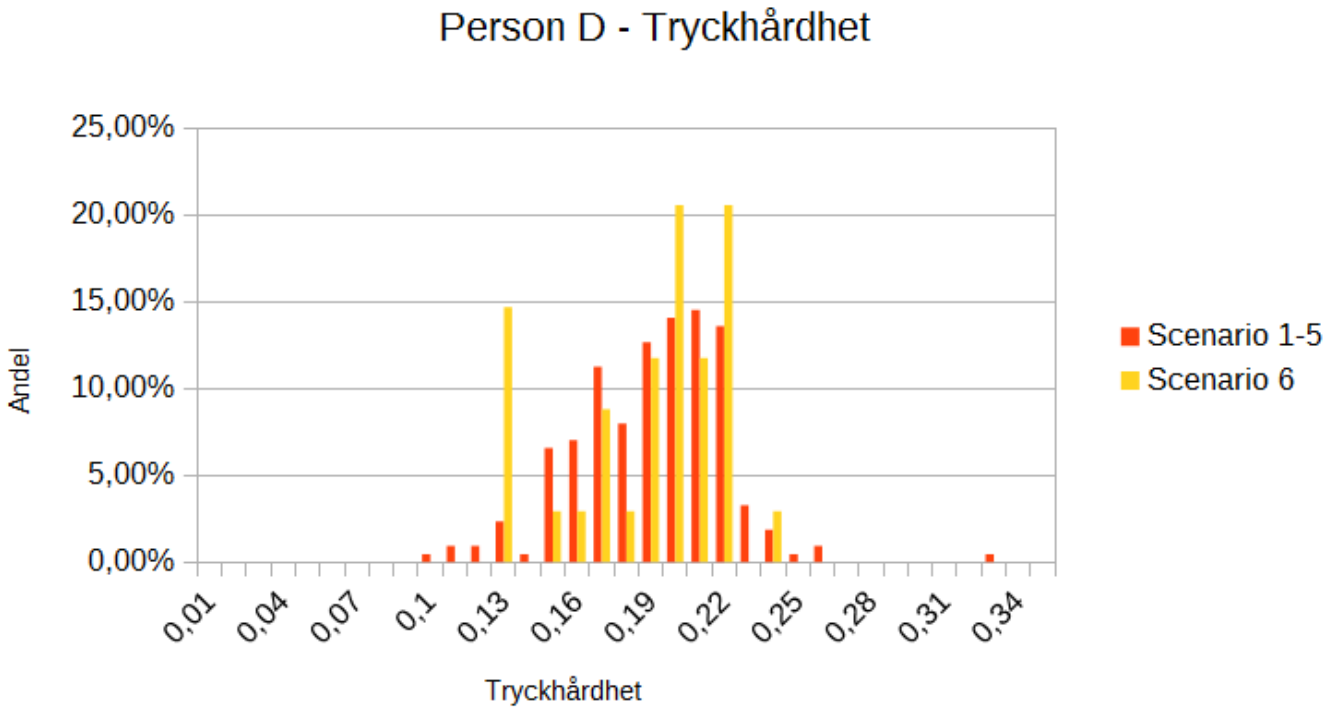
Scenario	Observationer	Avvikelser	Avvikelseratio	Frustration
1	82	2	2,4	0
2	148	15	10,1	0
3	98	8	8,2	0
4	36	5	13,5	0
5	69	6	8,7	0
6	102	12	11,8	0

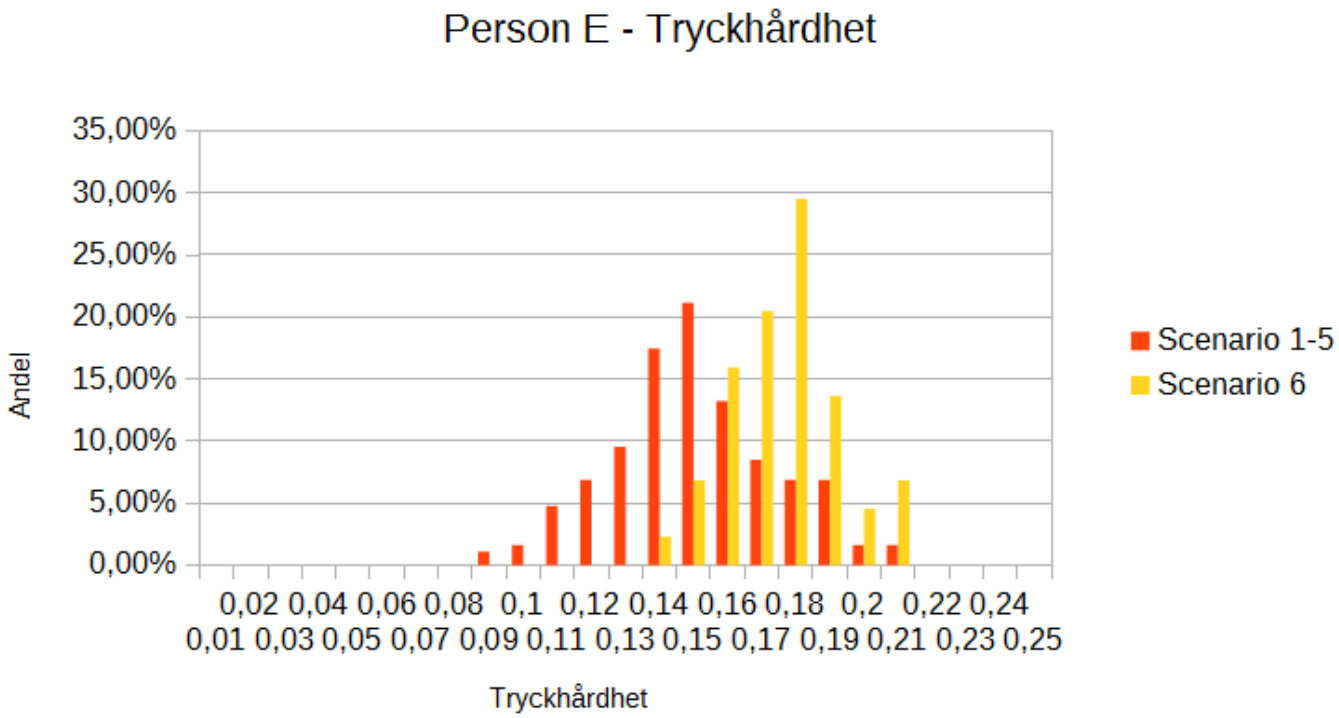
Bilaga F: Parametervärden

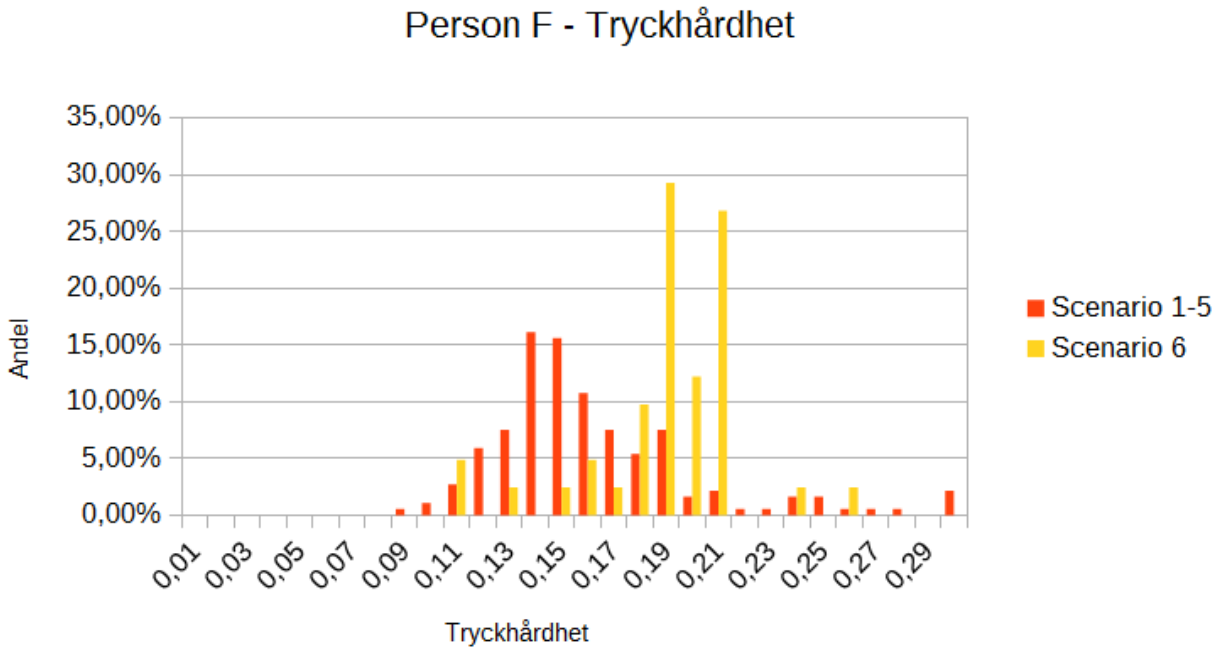
Tryckhårdhet



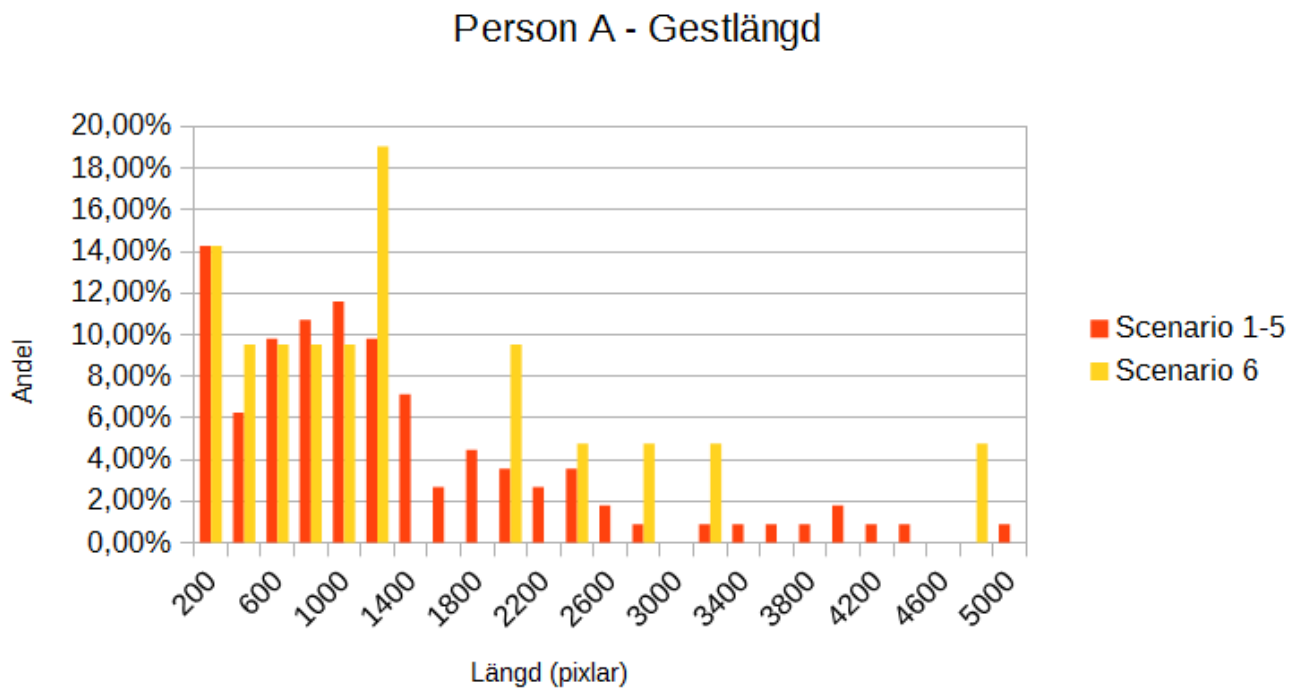


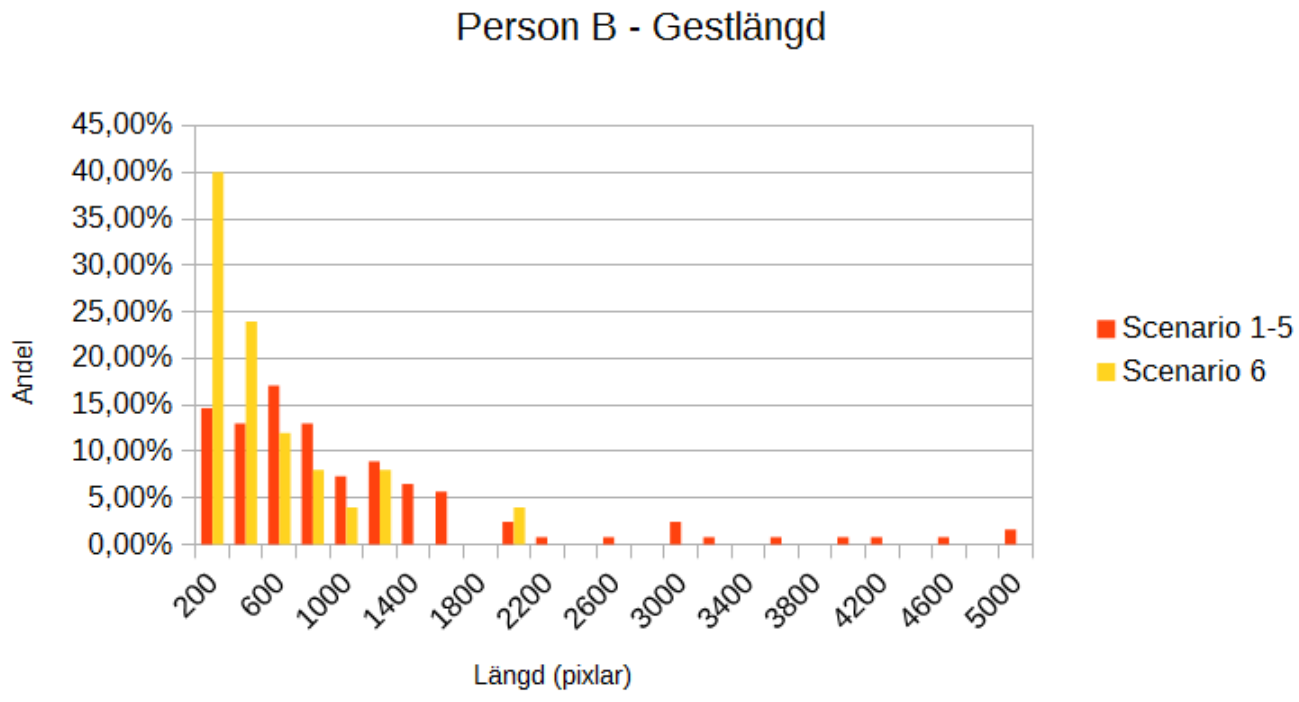

Person D - Tryckhårdhet
25,00%
20,00%
15,00%
10,00%
5,00%
0,00%
Andel
0,01
0,04
0,07
0,1
0,13
0,16
0,19
0,22
0,25
0,28
0,31
0,34
Tryckhårdhet
Scenario 1-5
Scenario 6

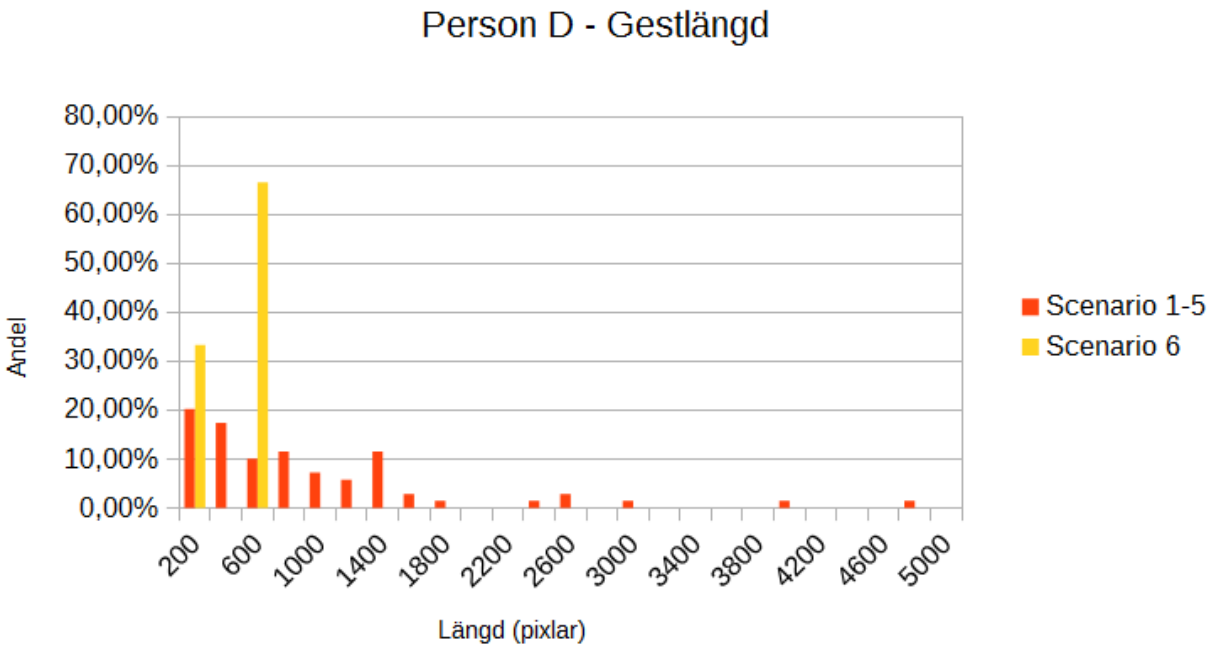

Person E - Tryckhårdhet
35,00%
30,00%
25,00%
20,00%
15,00%
10,00%
5,00%
0,00%
Andel
0,02 0,04 0,06 0,08 0,1 0,12 0,14 0,16 0,18 0,2 0,22 0,24
0,01 0,03 0,05 0,07 0,09 0,11 0,13 0,15 0,17 0,19 0,21 0,23 0,25
Tryckhårdhet
Scenario 1-5
Scenario 6

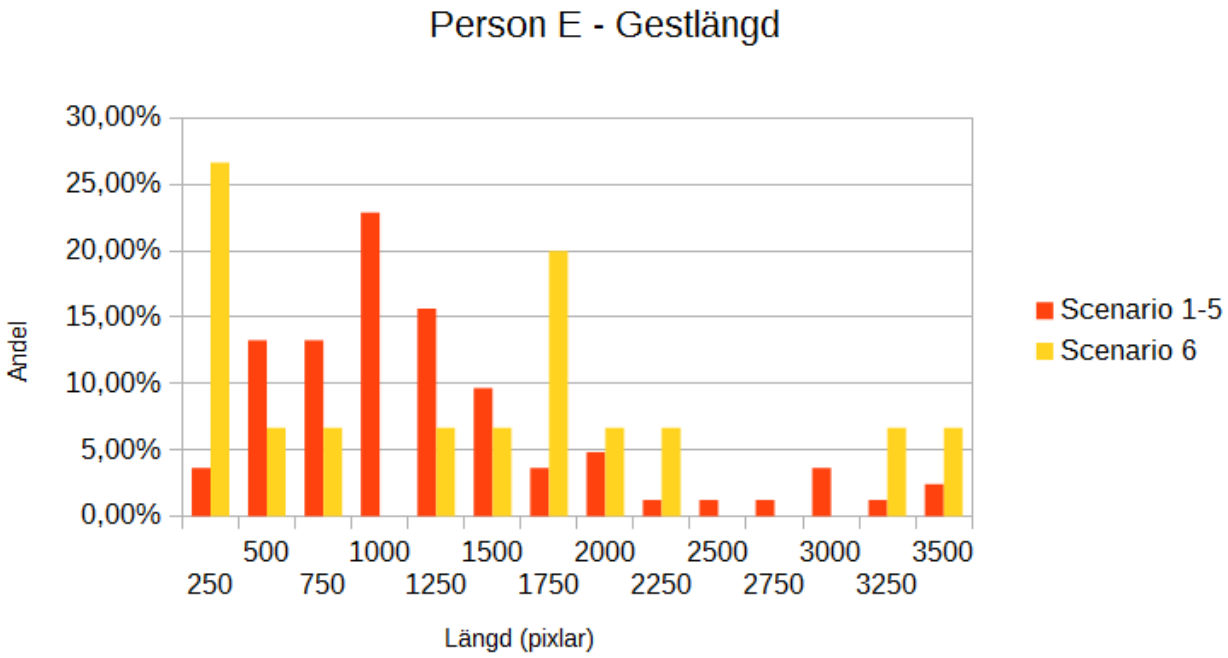

Person F - Tryckhårdhet
35,00%
30,00%
25,00%
20,00%
15,00%
10,00%
5,00%
0,00%
Andel
0,01
0,03
0,05
0,07
0,09
0,11
0,13
0,15
0,17
0,19
0,21
0,23
0,25
0,27
0,29
Tryckhårdhet
Scenario 1-5
Scenario 6

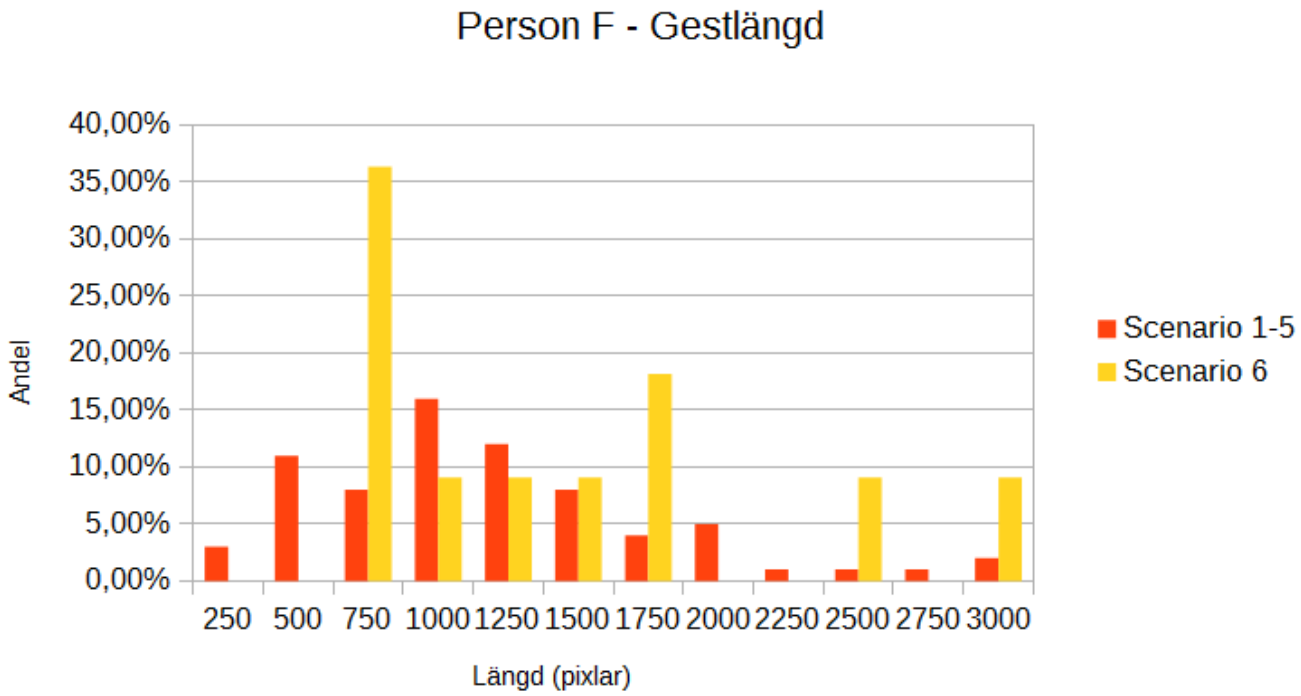
Gestlängd


Person A - Gestlängd
20,00%
18,00%
16,00%
14,00%
12,00%
10,00%
8,00%
6,00%
4,00%
2,00%
0,00%
Andel
200
600
1000
1400
1800
2200
2600
3000
3400
3800
4200
4600
5000
Längd (pixlar)
Scenario 1-5
Scenario 6

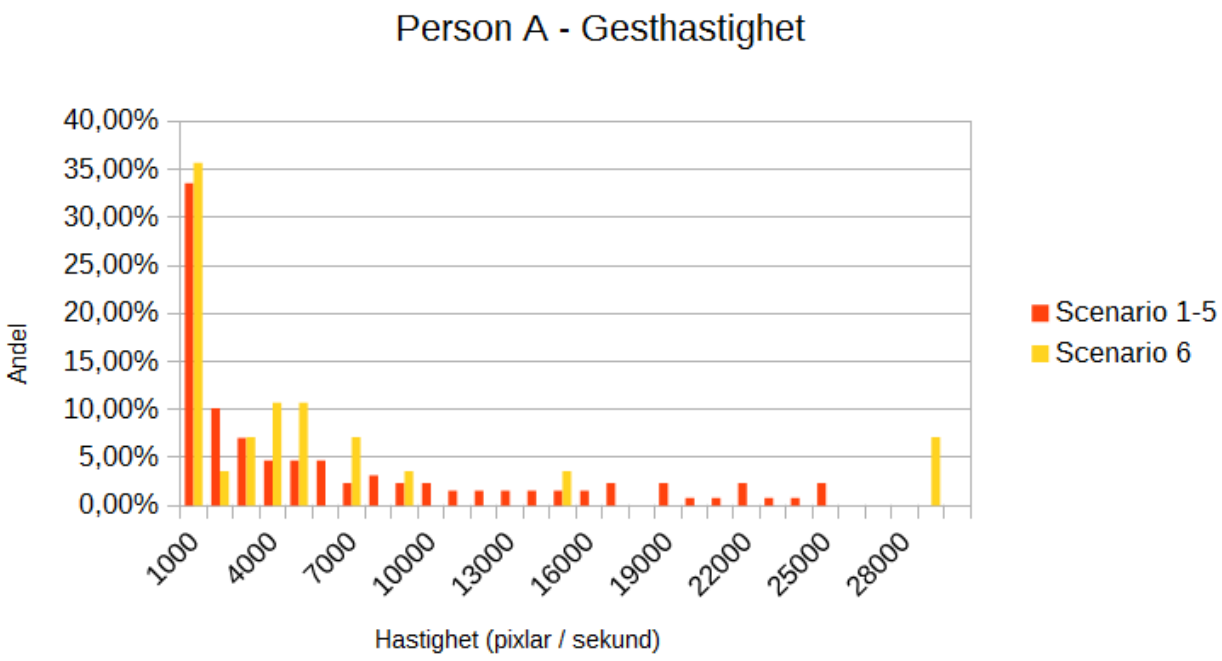

Person B - Gestlängd
45,00%
40,00%
35,00%
30,00%
25,00%
20,00%
15,00%
10,00%
5,00%
0,00%
Andel
200
600
1000
1400
1800
2200
2600
3000
3400
3800
4200
4600
5000
Längd (pixlar)
Scenario 1-5
Scenario 6

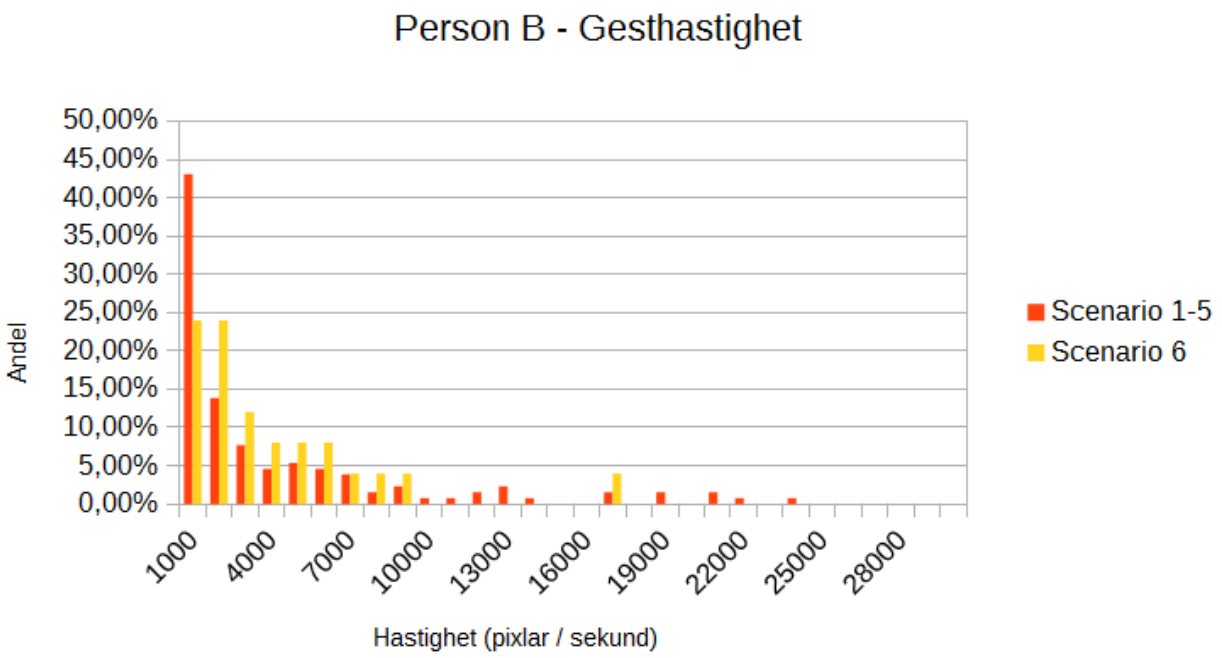

Person D - Gestlängd
80,00%
70,00%
60,00%
50,00%
40,00%
30,00%
20,00%
10,00%
0,00%
Andel
200
600
1000
1400
1800
2200
2600
3000
3400
3800
4200
4600
5000
Längd (pixlar)
Scenario 1-5
Scenario 6

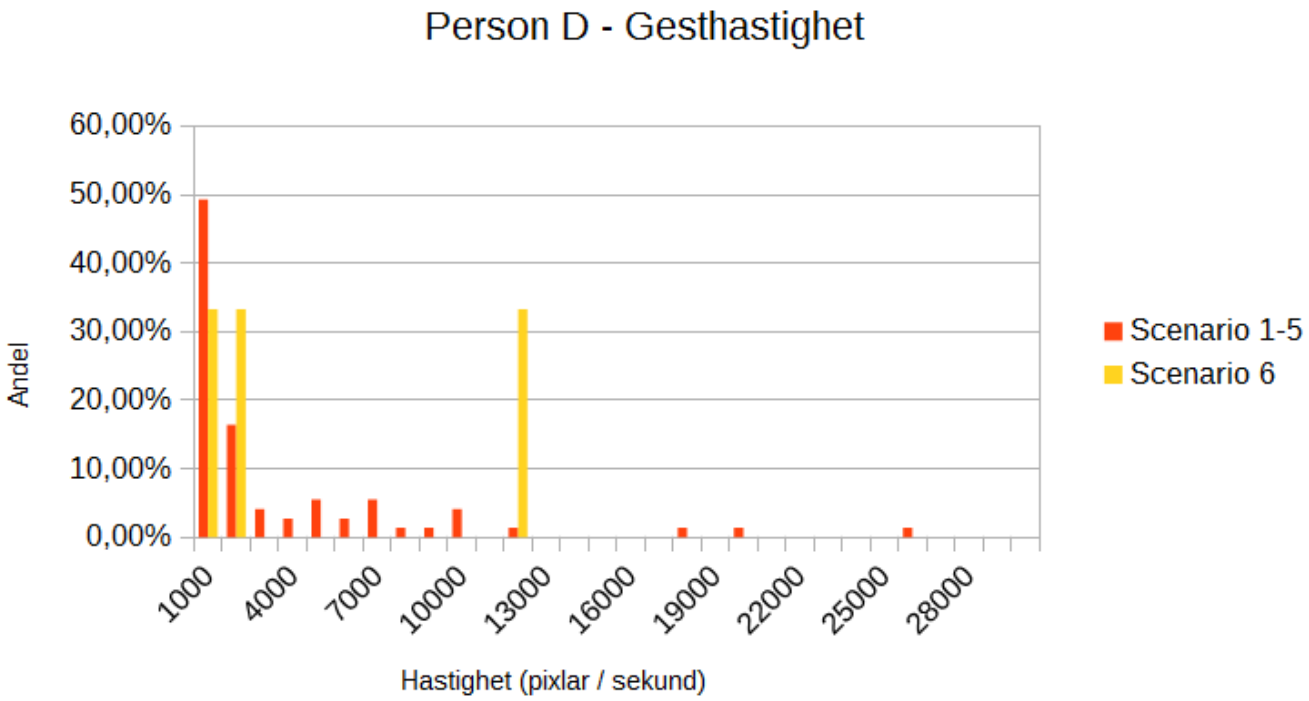

Person E - Gestlängd
30,00%
25,00%
20,00%
15,00%
10,00%
5,00%
0,00%
Andel
250
500
750
1000
1250
1500
1750
2000
2250
2500
2750
3000
3250
3500
Längd (pixlar)
Scenario 1-5
Scenario 6

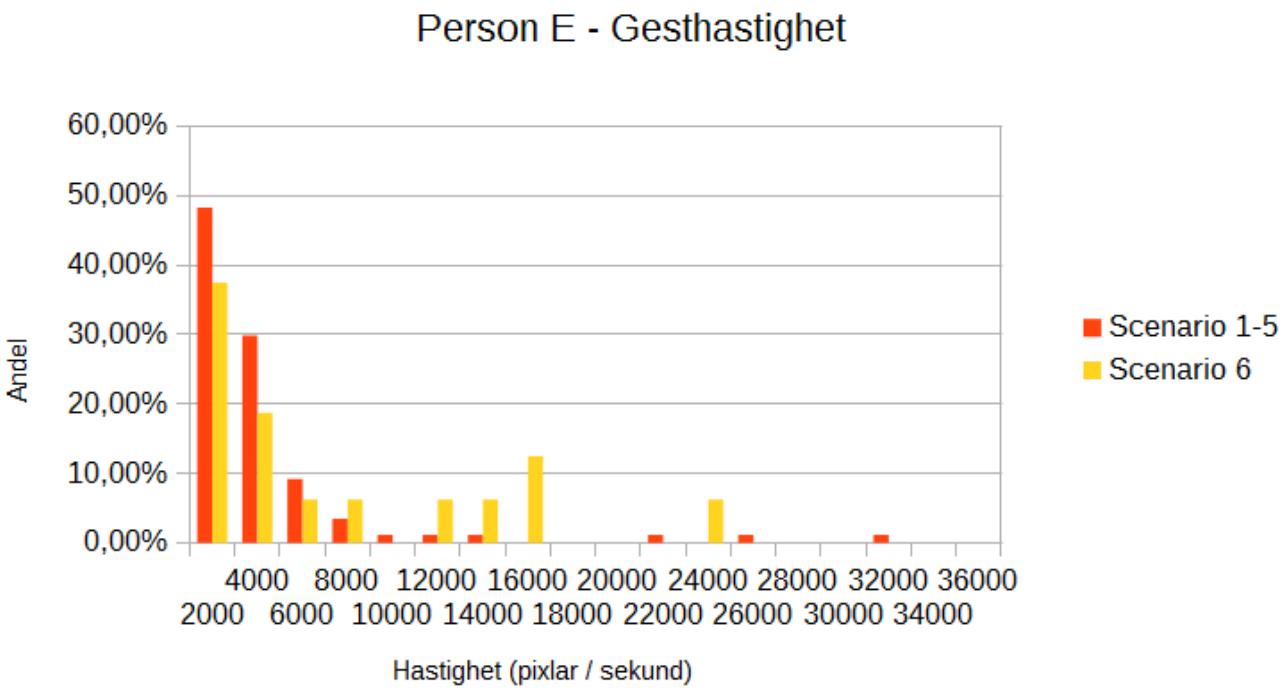

Person F - Gestlängd
40,00%
35,00%
30,00%
25,00%
20,00%
15,00%
10,00%
5,00%
0,00%
Andel
250
500
750
1000
1250
1500
1750
2000
2250
2500
2750
3000
Längd (pixlar)
Scenario 1-5
Scenario 6

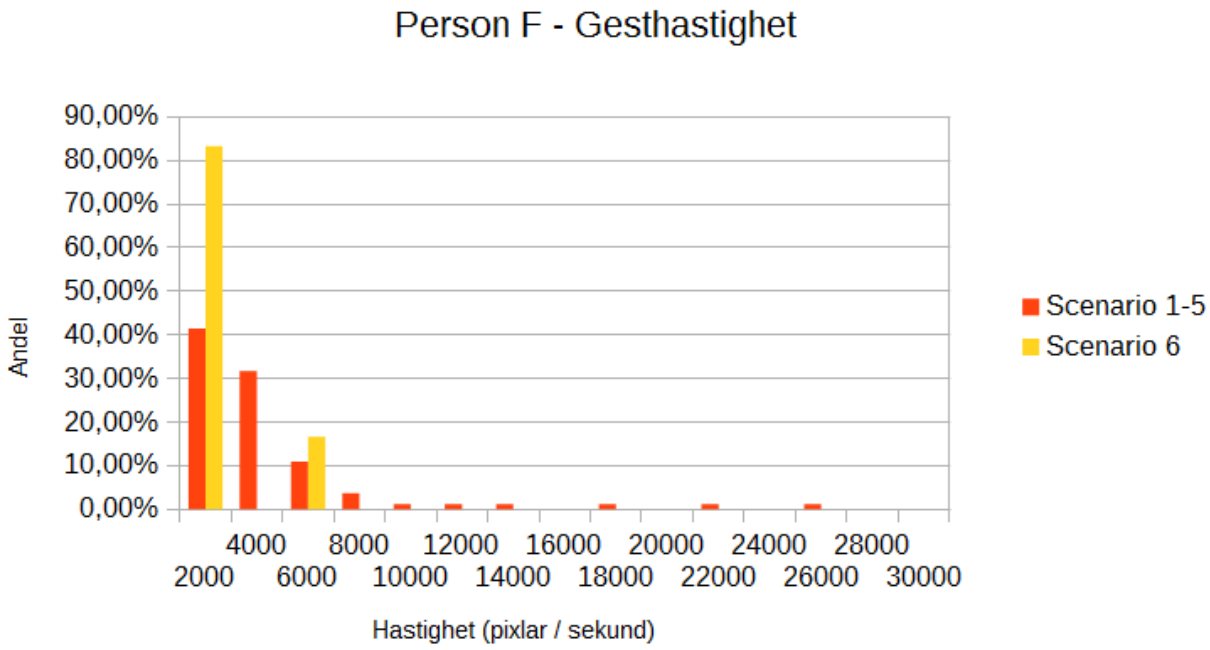
Gesthastighet



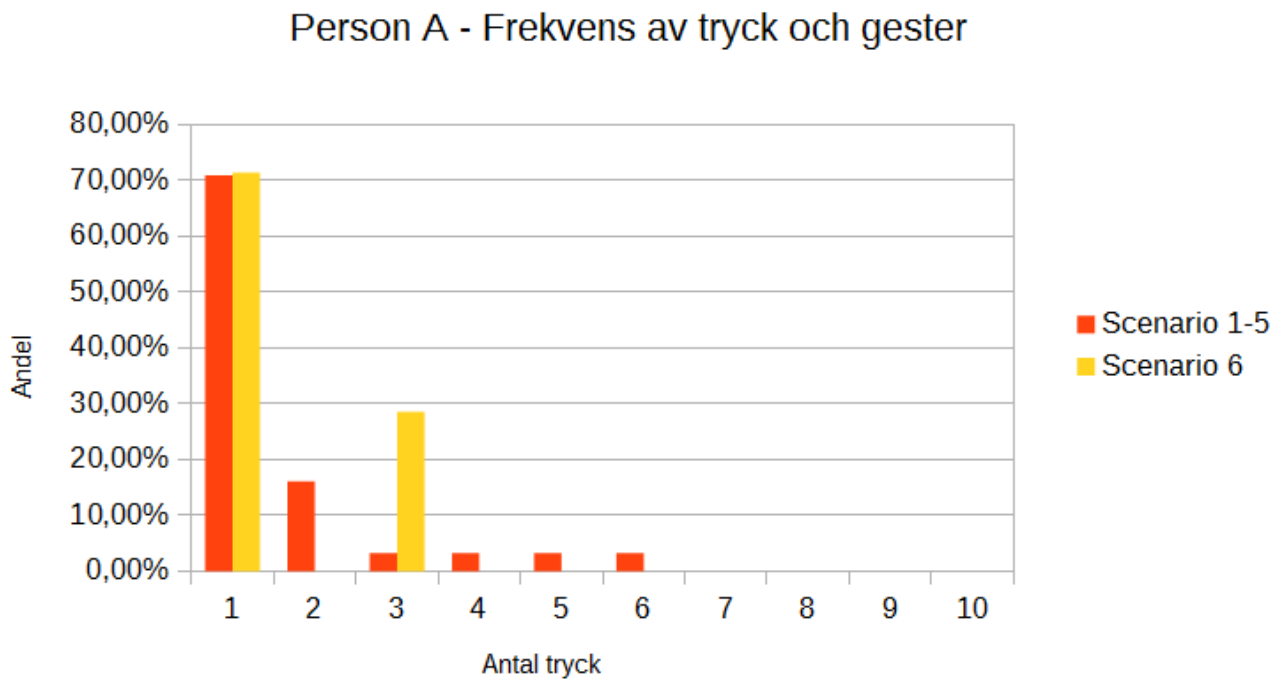


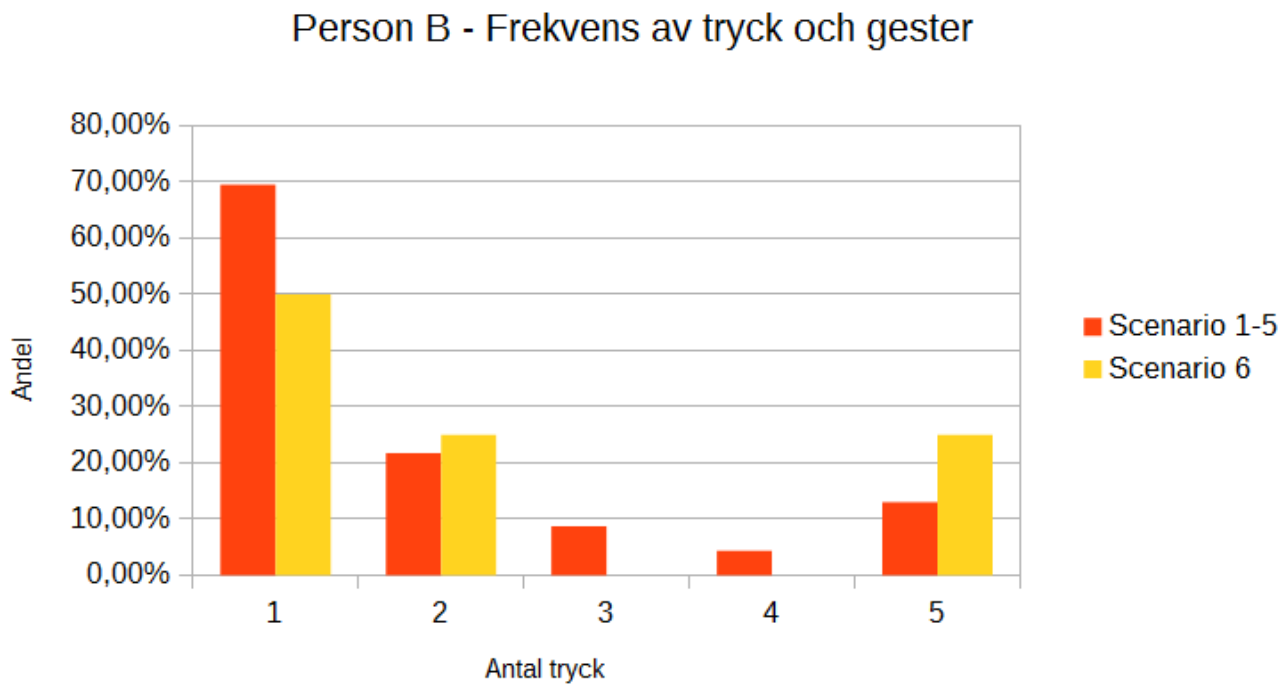

Person D - Gesthastighet
60,00%
50,00%
40,00%
30,00%
20,00%
10,00%
0,00%
Andel
1000
4000
7000
10000
13000
16000
19000
22000
25000
28000
Hastighet (pixlar / sekund)
Scenario 1-5
Scenario 6

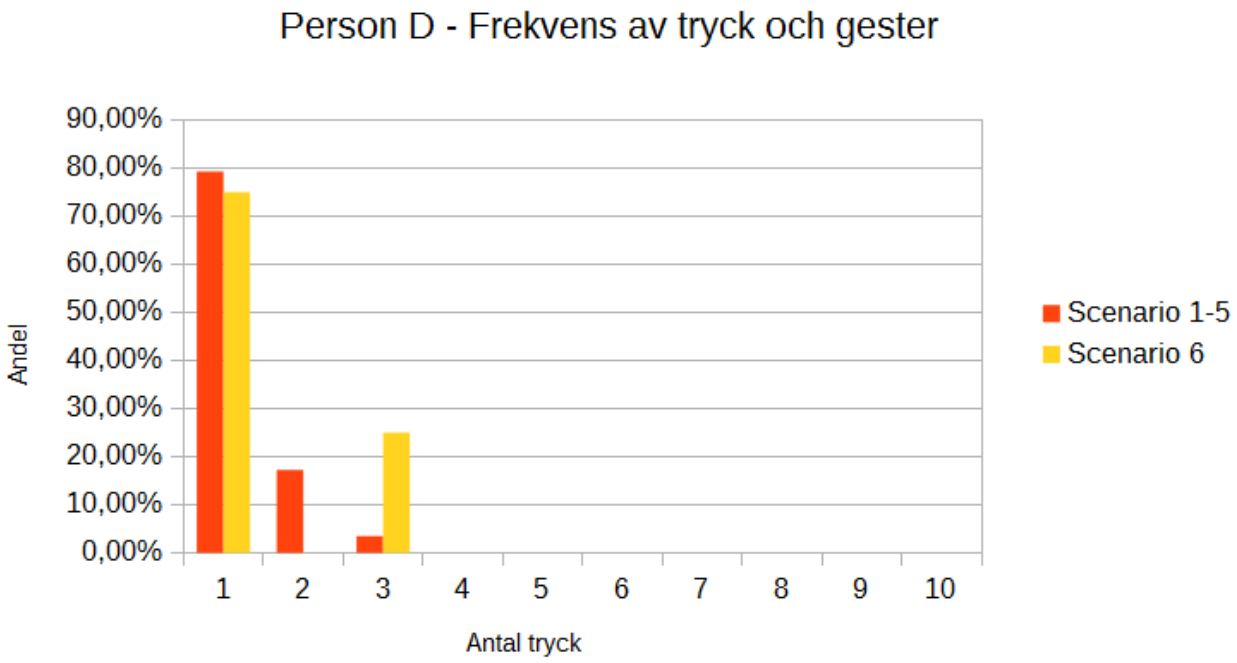

Person E - Gesthastighet
60,00%
50,00%
40,00%
30,00%
20,00%
10,00%
0,00%
Andel
2000
4000
6000
8000
10000
12000
14000
16000
18000
20000
22000
24000
26000
28000
30000
32000
34000
36000
Hastighet (pixlar / sekund)
Scenario 1-5
Scenario 6

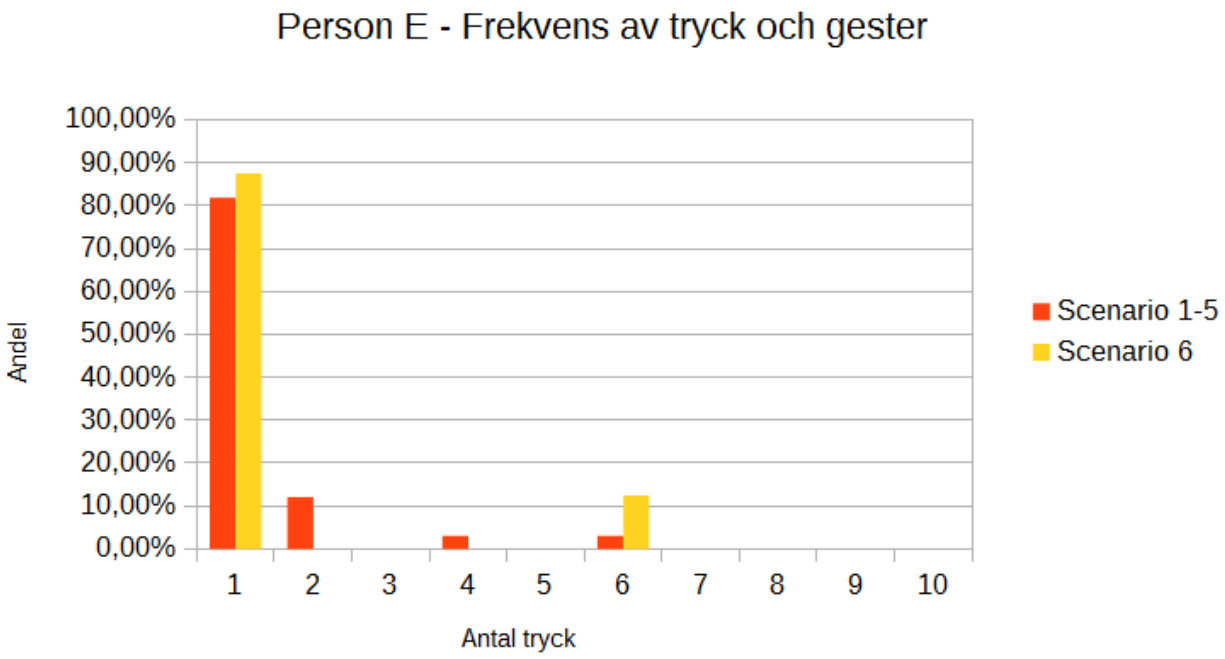

Person F - Gesthastighet
90,00%
80,00%
70,00%
60,00%
50,00%
40,00%
30,00%
20,00%
10,00%
0,00%
Andel
2000
4000
6000
8000
10000
12000
14000
16000
18000
20000
22000
24000
26000
28000
30000
Hastighet (pixlar / sekund)
Scenario 1-5
Scenario 6

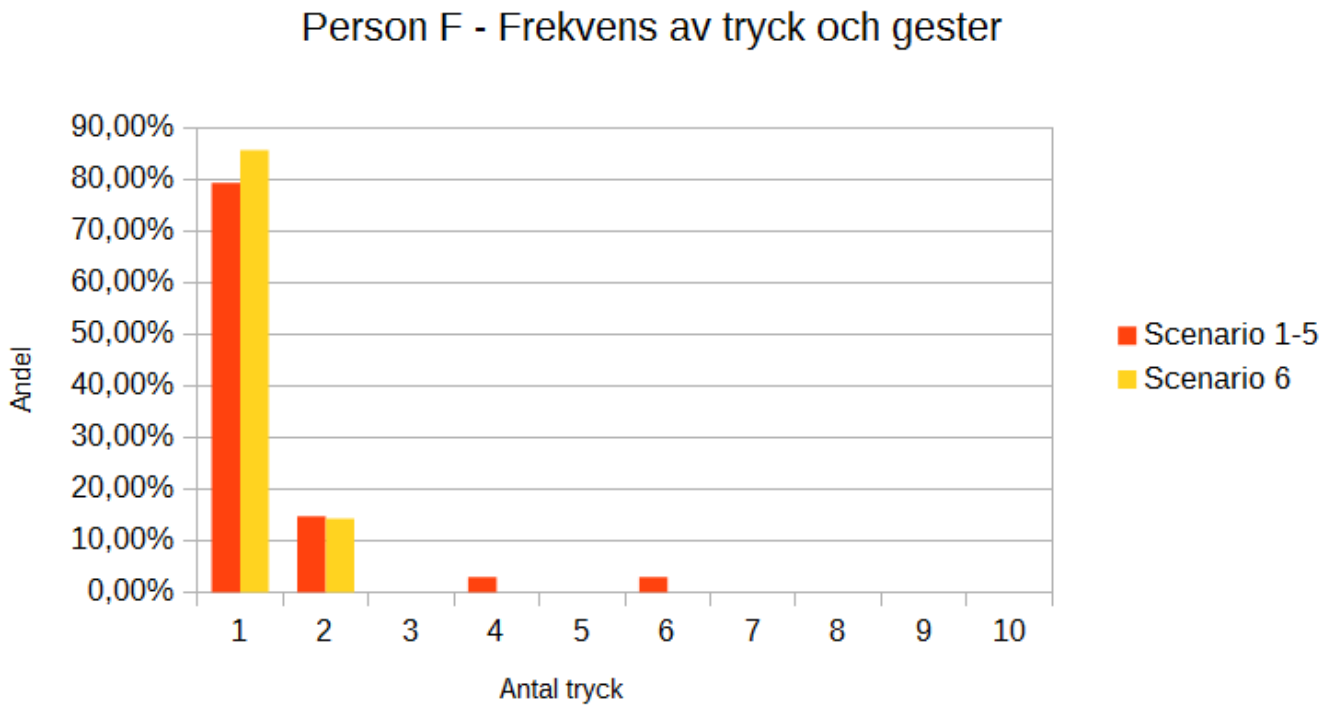
Frekvens av tryck och gester



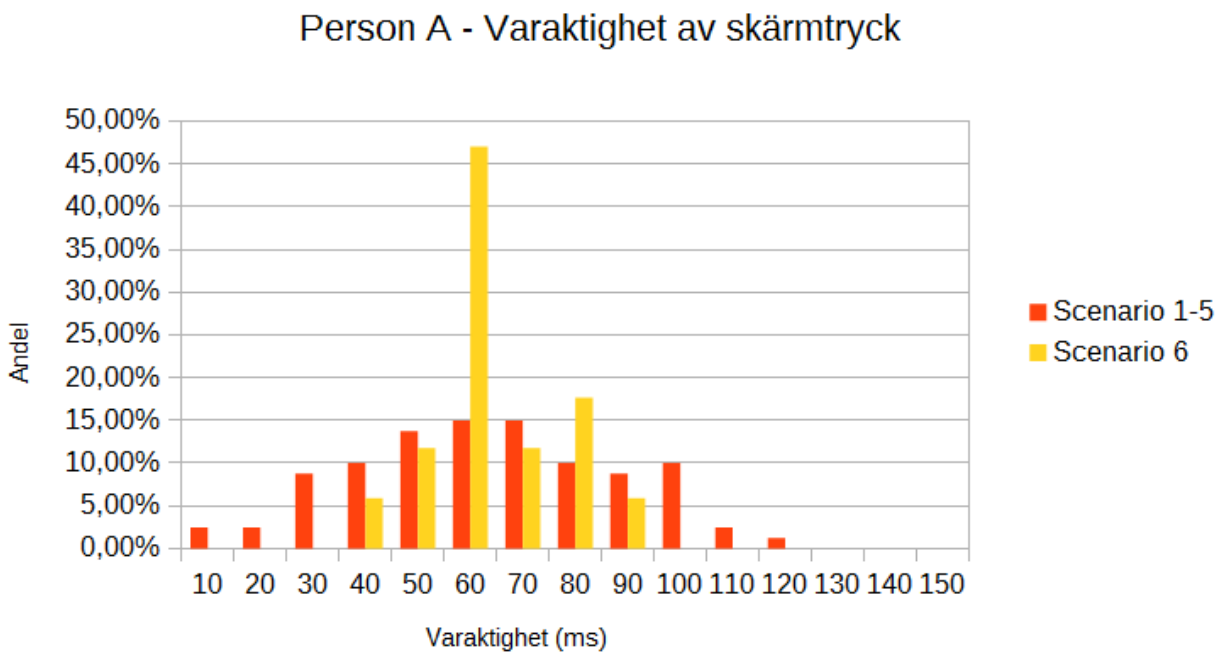


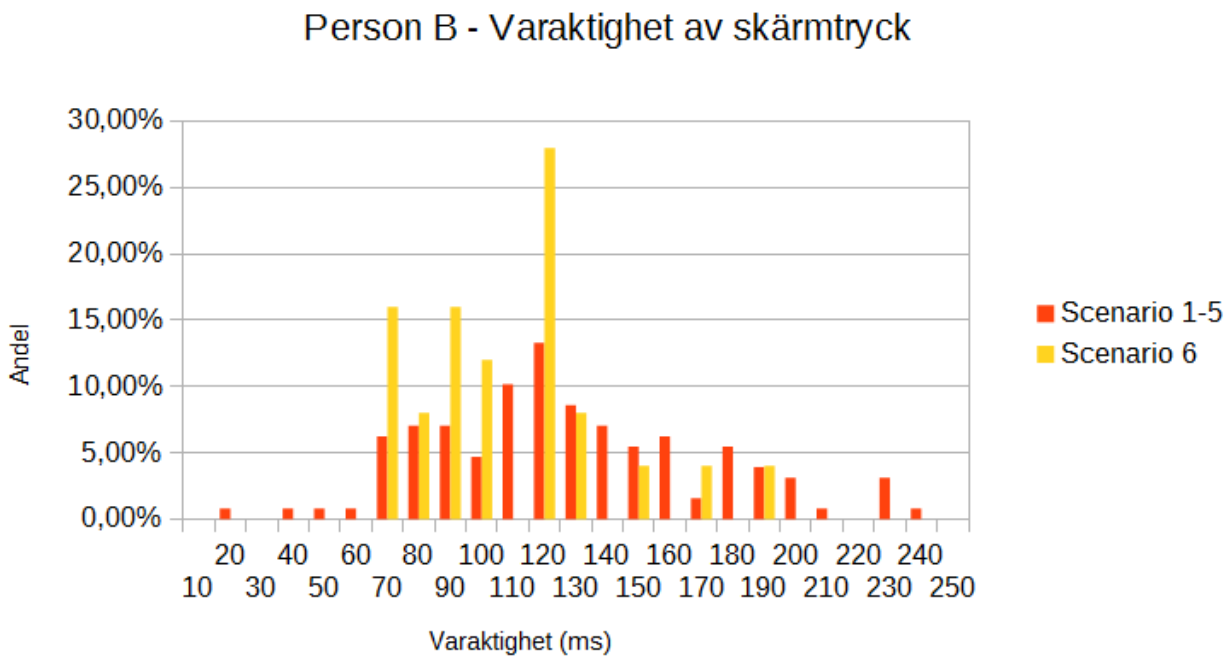

Person D - Frekvens av tryck och gester
90,00%
80,00%
70,00%
60,00%
50,00%
40,00%
30,00%
20,00%
10,00%
0,00%
Andel
1
2
3
4
5
6
7
8
9
10
Antal tryck
Scenario 1-5
Scenario 6

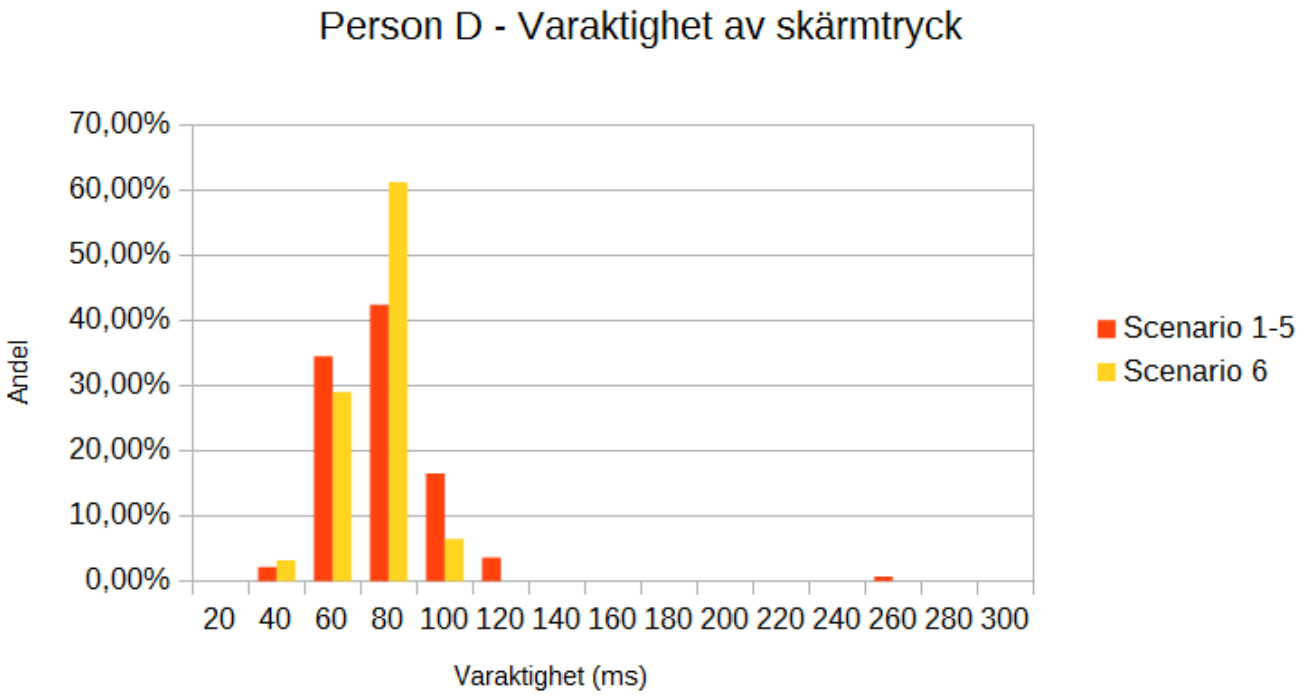

Person E - Frekvens av tryck och gester
100,00%
90,00%
80,00%
70,00%
60,00%
50,00%
40,00%
30,00%
20,00%
10,00%
0,00%
Andel
1
2
3
4
5
6
7
8
9
10
Antal tryck
Scenario 1-5
Scenario 6

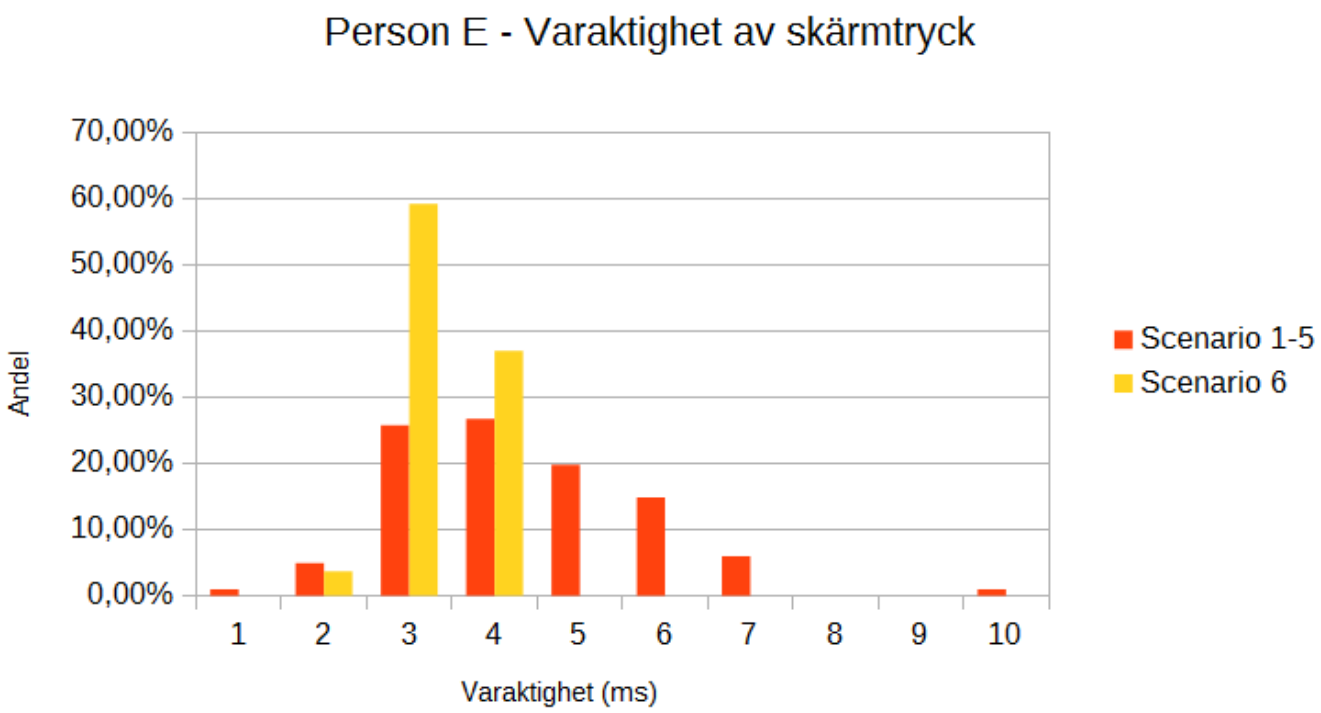

Person F - Frekvens av tryck och gester
90,00%
80,00%
70,00%
60,00%
50,00%
40,00%
30,00%
20,00%
10,00%
0,00%
Andel
1
2
3
4
5
6
7
8
9
10
Antal tryck
Scenario 1-5
Scenario 6

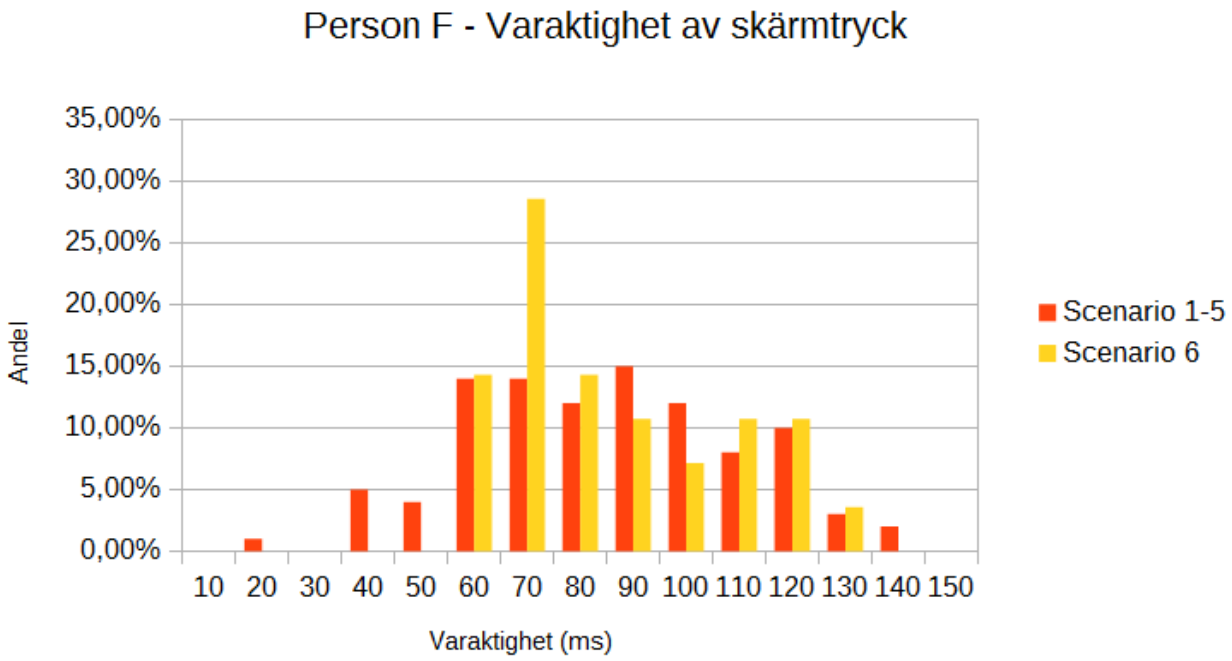
Varaktighet av skärmtryck



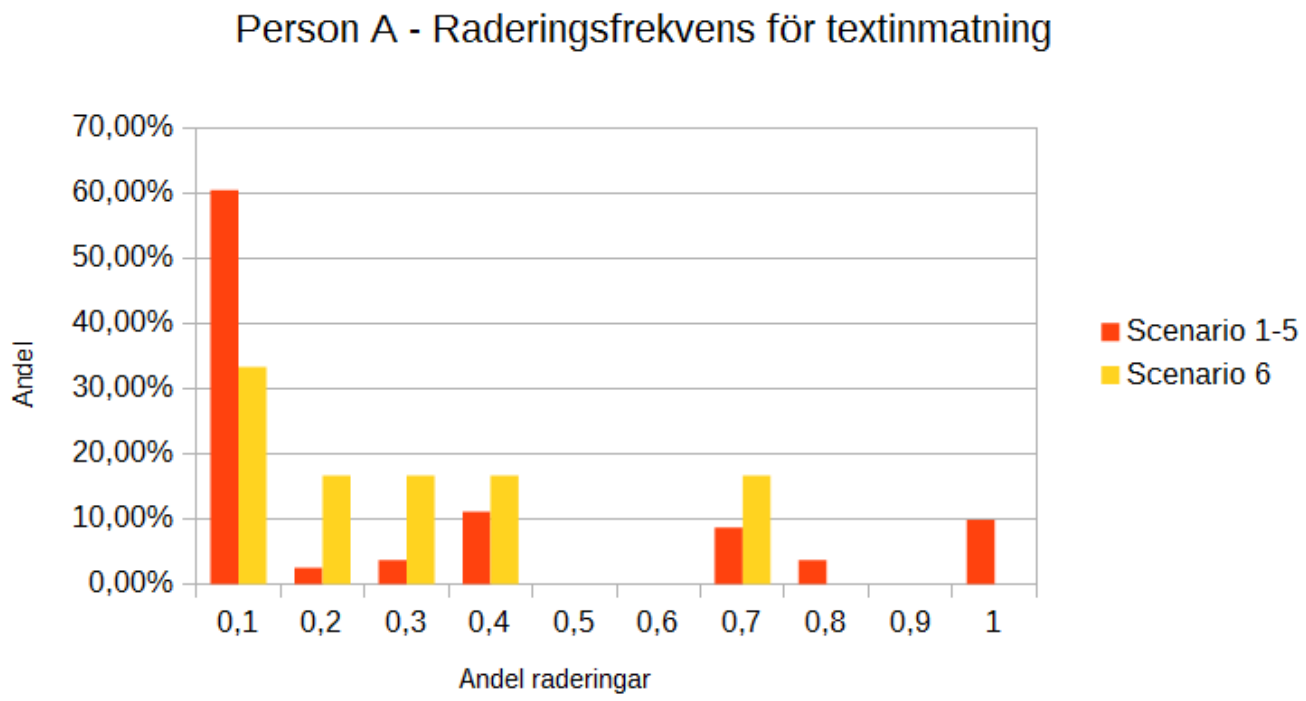


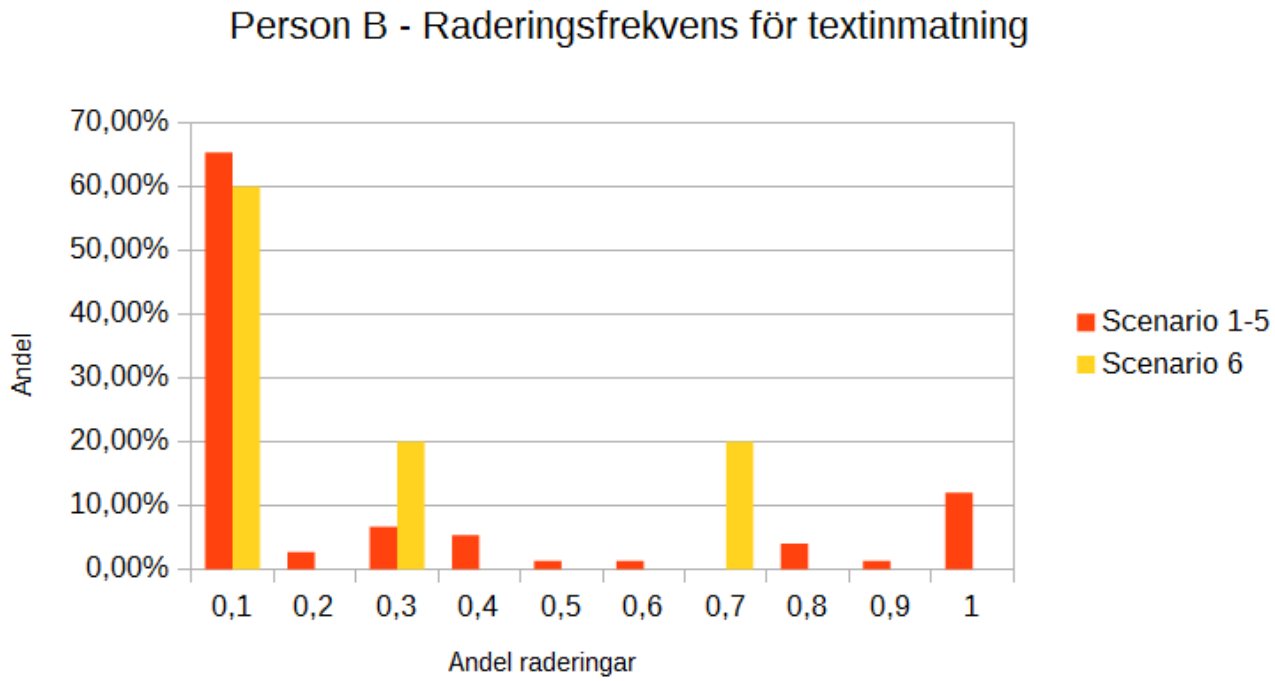

Person D - Varaktighet av skärmtryck
70,00%
60,00%
50,00%
40,00%
30,00%
20,00%
10,00%
0,00%
Andel
20 40 60 80 100 120 140 160 180 200 220 240 260 280 300
Varaktighet (ms)
Scenario 1-5
Scenario 6

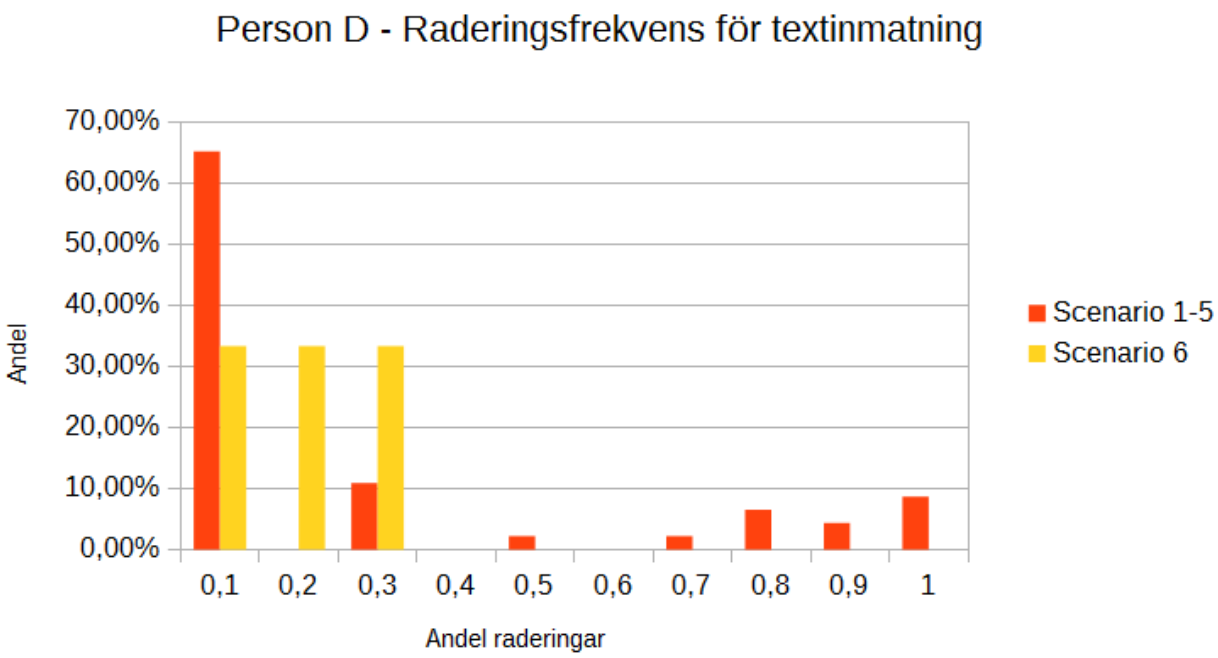

Person E - Varaktighet av skärmtryck
70,00%
60,00%
50,00%
40,00%
30,00%
20,00%
10,00%
0,00%
Andel
1 2 3 4 5 6 7 8 9 10
Varaktighet (ms)
Scenario 1-5
Scenario 6

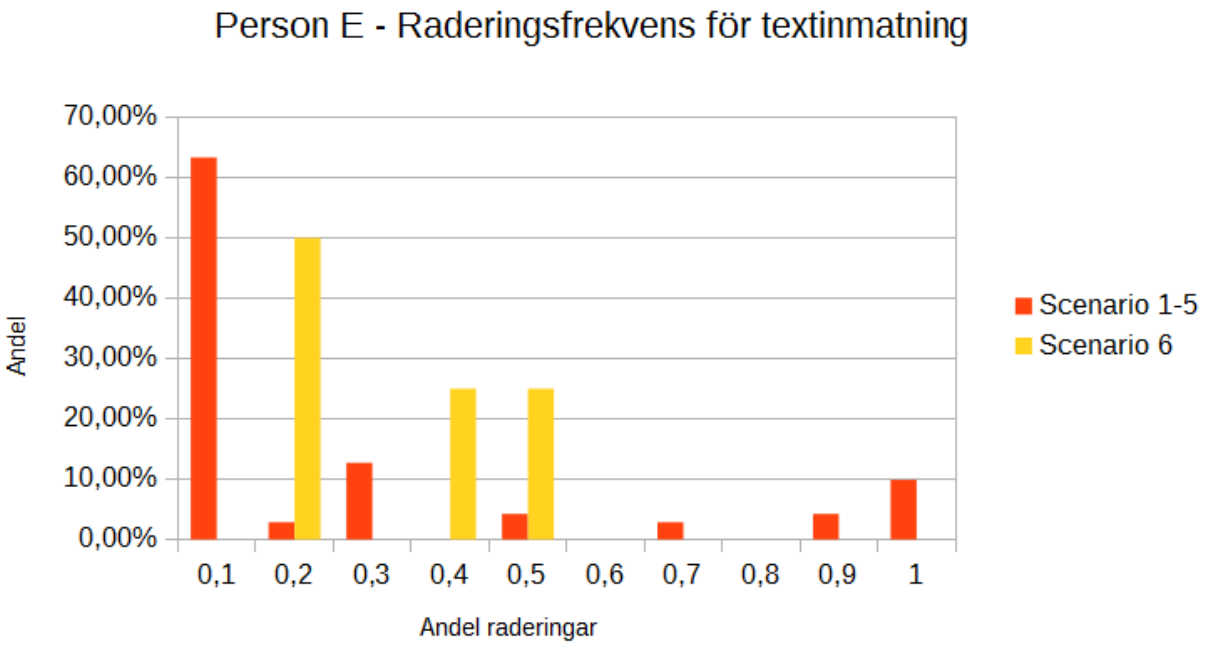

Person F - Varaktighet av skärmtryck
35,00%
30,00%
25,00%
20,00%
15,00%
10,00%
5,00%
0,00%
Andel
10 20 30 40 50 60 70 80 90 100 110 120 130 140 150
Varaktighet (ms)
Scenario 1-5
Scenario 6

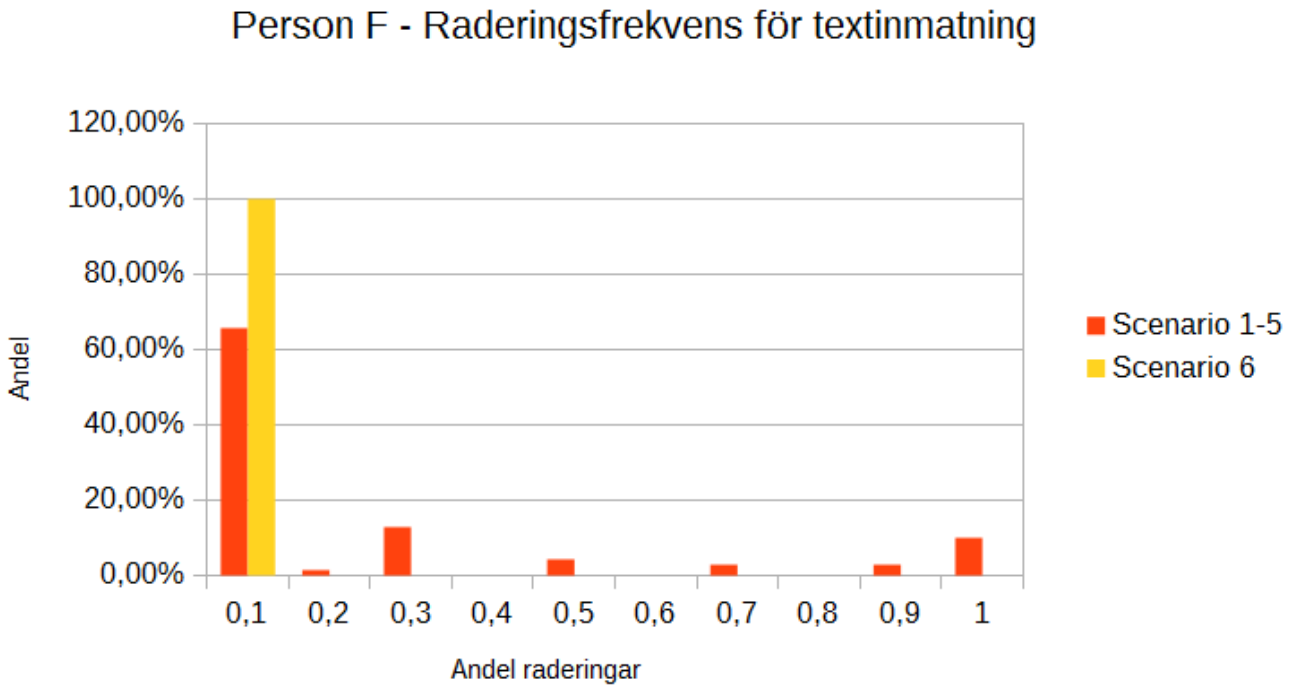
Raderingsfrekvens för textinmatning






Person D - Raderingsfrekvens för textinmatning
70,00%
60,00%
50,00%
40,00%
30,00%
20,00%
10,00%
0,00%
Andel
0,1
0,2
0,3
0,4
0,5
0,6
0,7
0,8
0,9
1
Andel raderingar
Scenario 1-5
Scenario 6


Person E - Raderingsfrekvens för textinmatning
70,00%
60,00%
50,00%
40,00%
30,00%
20,00%
10,00%
0,00%
Andel
0,1
0,2
0,3
0,4
0,5
0,6
0,7
0,8
0,9
1
Andel raderingar
Scenario 1-5
Scenario 6


Person F - Raderingsfrekvens för textinmatning
120,00%
100,00%
80,00%
60,00%
40,00%
20,00%
0,00%
Andel
0,1
0,2
0,3
0,4
0,5
0,6
0,7
0,8
0,9
1
Andel raderingar
Scenario 1-5
Scenario 6