UPPSALA
UNIVERSITET

TVE 13 002 januari
Examensarbete 15 hp
Januari 2013

Loggning av användares interaktioner

utvärdering av användbarhet

Carl Gleisner

Institutionen för teknikvetenskaper
Department of Engineering Sciences

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Loggning av användares interaktioner för utvärdering av användbarhet

Logging user interactions for usability analysis

Carl Gleisner

The disconnect between assumptions regarding computer users interactions made by software engineers and the actual interactions do, for various reasons, not always match. Gathering actual usage data from user interfaces can enable software designers to make better educated decisions. By buildning a prototype for collecting such data from a diagnosis tool for the automotive industry this project has shown not only how to benefit from the data but also how to perform the work. The data is 1 600 sessions consisting of 26 000 individual events. From this it has shown that available tools can assist in rudimentary questions regarding usage. For more elaborate studies specialized or custom tools must be acquired or developed.

Handledare: Jan Svensson, Scania CV AB
Ämnesgranskare: Iordanis Kavathatzopoulos
Examinator: Nóra Masszi
UPTEC FRIST** ***

Sammanfattning

Särkopplingen mellan antganden mjukvaruutvecklare gör om datoranvändares interaktioner och hur interaktionerna sker i den verkliga användningen stämmer inte alltid överens. Genom att samla data om interaktioner med gränssnittet kan utvecklare och formgivare fatta bättre underrättade beslut. Genom att bygga en prototyp för att samla in sådan data från ett diagnosverktyg i automotivebranschen har detta projekt visat inte enbart hur nytta kan dras från sådan data utan även hur det görs rent tekniskt. Datan utgörs av 1 600 sessioner bestående av 26 000 enskilda händelser. Från detta har det visat sig möjligt att dra slutsatser med allmänt spridda verktyg. För mer utförliga studier krävs dels speciella analysverktyg, dels större arbete om vilka slutsatser som är rimliga att dra från analysen.

Innehåll

Figurer

Tabeller

Vissa förkortningar och akronymer

GUI Graphical User Interface

ISO International Organization for Standardization

SDP3 Scania Diagnos och Programmer

SQL Structured Query Language

WCF Windows Communication Foundation

WPF Windows Presentation Framework

1 Inledning

1.1 Bakgrund

En klassisk utmaning inom mjukvaruutveckling är att minska distansen mellan mjukvaruingenjören och användarens uppfattning om hur en applikation ska användas. Eftersom utvecklaren endast i begränsad utsträckning kan förutse användarens uppfattningar om detta nödgas hon göra antaganden baserade på sina egna förväntningar. Utvecklarens förväntningar kommer från dennes kunskaper om kravspecifikationer, tidigare erfarenheter i systemutveckling, kunskaper på det aktuella området, kunskaper om specifika uppgifter och från egna erfarenheter av att använda applikationer. Då utvecklaren gör antaganden som inte stämmer överens med användarnas uppfattningar försämras användbarheten.[3]

Användbarhet har bland annat definierats enligt nedan.

> *"Utsträckningen i vilken en produkt kan användas av specificerade användare för att uppnå specificerade mål på ett ändamålsenligt, effektivt sätt och med tillfredsställelse i ett specificerat sammanhang."*[1]

I syfte att förbättra användbarheten kan ingenjören skaffa sig ett bättre beslutsunderlag. Detta kan görs genom att utföra s.k. användbarhetstestning. Användbarhetstestning associeras traditionellt med att i laboratoriemiljö testa en mer eller mindre slutförd produkt på representativa användare. Sådana metoder ger undersökaren möjlighet att direkt observera deltagarna i testen och föra en dialog med dem. Denna inblandning riskerar dock att kontaminera resultaten.

Ett alternativt sätt att studera användbarhet är att istället registrera användarnas faktiska interaktioner med gränssnittet i stor skala och sedan analysera den insamlade datan och dra slutsatser därefter. Lockelsen med tillvägagångssättet är bl.a. den potentiella kostnadseffektiviteten och möjligheten till förbättrad statistisk tillförlitlighet. Vidare kan metoden tänkas ge ledning i frågor där de traditionella metoderna brister, exempelvis vilka funktionaliteter som inte används.[8]

[1] ISO 9241-11, egen översättning.

1.2 Problem

Att genomföra sådana tester med registrering av interaktioner och insamling innebär ett antal utmaningar. Applikationen måste manipuleras och denna s.k. *instrumentering* kommer oundvikligen att ha viss inverkan. Här kan ett antal val göras vilka innebär både för- och nackdelar. Givna utgångspunkter måste vara att förändringen i sig inte inför omotiverade beroenden eller nya fel.

Utöver instrumenteringens inverkan bör det även övervägas *vad* som ska registreras och vad det i sin tur *representerar*. De frågor som måste besvaras är huruvida det går det att lita på att datan som samlats in är tillräckligt noggrann samt om datan är representativ givet de ställda frågorna. Dessa frågor ger svar på vad som inom vetenskapsteorin kallas mätningarnas *reliabilitet* respektive *validitet*. Reliabiliteten anger hur mätningarnas tillförlitlighet, medan validiteten anger i vilken utsträckning de faktiskt ger uttryck för det som avsågs att mätas.[9]

Den sista utmaningen är att analysera den insamlade datan på ett sätt som är praktiskt och kostnadseffektivt i den aktiva utvecklingsorganisationen. Vidare måste analysen uppfylla vissa krav så som att den inte påverkar datan på ett sätt som förvränger resultatet utan ger något nyttigt och användbart. Mot denna bakgrund är det projektets mål att behandla följande tre frågor.

Q1 Hur ska instrumenteringen utföras på ett ingenjörsmässigt korrekt sätt?

Q2 Vilken data är relevant givet de ställda frågorna?

Q3 Hur ska datan effektivt analyseras för mest rättvisande bild?

Även att alla de tre frågorna berörs vilar projektets huvudsakliga fokus på den tredje frågan. Detta motiveras av att det är den mest intressanta rent vetenskapligt och samtidigt mest utmanande för mjukvaruingenjören som kanske endast tillfälligt antar rollen som användbarhetsingenjör.

1.3 Syfte

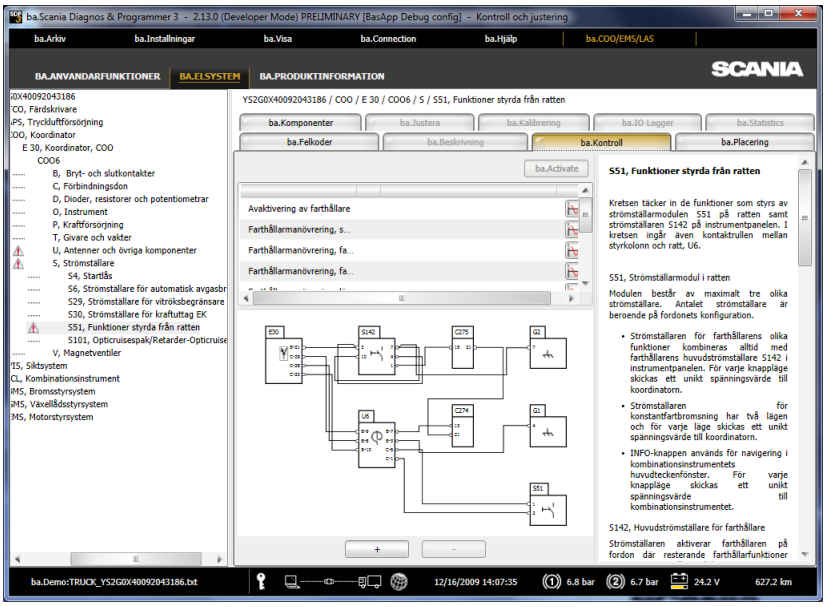
De för projektet utvalda frågorna syftar till att ge mjukvaru- och användbarhetsingenjören en god uppfattning om dels vilka frågor loggning av användarinteraktioner kan ge ledning i, dels hur dylika tester utförs på ett rationellt sätt i den aktiva organisationen. Med rationellt avses att det är kostnadseffektivt och att resultatet är förbättrad användbarhet.

Eftersom det inte är rimligt att inom de givna ramarna införa en instrumentering i produktionsmiljö avgränsas arbetet till däremot omfatta en en prototyp av en sådan instrumentering. Förstudien och teoriöversikten fokuserar på de tillvägagångssätt som kan tänkas ha bäring på prototypens tillämpning.

1.4 Bolaget och verktyget

Projektet utförs hos Scania CV AB i Södertälje på avdelningen System Development Scania Diagnos And Testing [*sic*] (YSEI). Scania är en global fordonstillverkare med verksamhet i ungefär 100 länder och har ca. 37 500 anställda. Av dessa arbetar ca. 15 000 med försäljning och tjänster i bolagets egna dotterbolag samt ca. 2 900 inom forskning och utveckling. Produkterna är främst lastbilar men även bussar och motorer till industrin och marinbranschen. År 2011 omsatte bolaget 87,7 mdkr.[1]

För service och reparation av fordonen använder reparatören det egenutvecklade verktyget Scania Diagnos och Programmer 3 [*sic*], förkortat SDP3. Verktyget består av en Windows-applikation som installeras på en vanlig bärbar dator och ansluts till fordonet med en specialtillverkad kabel. SDP3 kommunicerar med fordonets 5 - 35 styrenheter (servrar) och läser av en mängd värden. Om en styrenhet rapporterar ett fel ges reparatören stöd i hur det ska åtgärdas.



Figur 1: *Huvudfönstret i applikationen.*

Det stöd reparatören får av SDP3 ska ge honom en funktionsbeskrivning, ett elschema och tillhörande toleransvärden. Han kan även få rapporter om trasiga komponenter eller få ledning i hur dessa ska kalibreras. I vissa fall kan även komponenterna kalibreras med hjälp av signal från SDP3. Innehållet i applikationen kommer från konstruktörer och s.k. metodingenjörer. Det är metodingenjörerna som tar fram de rekommenderade tillvägagångssätten för att åtgärda givna fel.

2 Litteraturstudie

2.1 Terminologi

Tre centrala begrepp kommer användas för att klassificera datan och definieras nedan.

Session avser tillfället då en användare interagerar med en instans av applikationen.

Händelse är den minsta enheten för att representera en interaktion i gränssnittet. Det kan vara exempelvis en knapptryckning, mustryckning eller menyval. Vad som avses här gäller *konkreta* händelser.

Händelseström är hela mängden av händelser som observerats under en session.

2.2 Användbarhetstestning

Som redan nämnts erbjuder det internationella organet ISO en definition på *användbarhet* (se 1.1). Så kallad användbarhetstestning har Lewis (2006) definierat som att generellt involvera representativa användare som ger sig an representativa uppgifter i representativa miljöer på tidiga prototyper av användargränssnitt för datorer. Testningen kan involvera allt från papperslappar på ett bord till sjösatta produkter i produktionsmiljö.

En viktig skillnad är den mellan forskning och testning avseende användbarhet. Den grundläggande skillnaden ligger i de båda aktiviteternas mål. Användbarhetstestning söker felaktigheter i gränssnittets användbarhet i syfte att förbättra denna. Samtidigt bör det som fungerar väl upptäckas och därefter kunna bevaras. Av vikt är att metoderna ska vara praktiska och ha stor påverkan på produkten och dess användbarhet . Utöver detta finns även fältet forskning kring metoder för användbarhetstestning [8].

Nedan sammanställs en del av de vanligaste metoderna för att utvärdera användbarhet.[4]

Expertutvärdering Vid en expertutvärdering anlitas en person som besitter betydande erfarenhet och kunskap på området användbarhet. Denne granskar produkten och identifierar egenskaper som kan leda till försämrad användbarhet. En fördel med denna metod är att testningen i sig tar kort tid att utföra. Det bör dock noteras att experten sannolikt inte är bekant med användarnas uppgifter och den miljön de arbetar i.

Fältstudier innebär att en grupp av användare observeras i sina faktiska arbetsmiljöer. Den här typen av studier genomförs vanligtvis innan större omarbetningar i produkten görs. En mindre grupp av användare väljs ut och får sedan svara på en enkät med bakgrundsfrågor. Därefter får de ge sin syn på den studerade produkten.

En fördel med metoden är att den utförs i en realistisk miljö och därför ger en inblick i faktiska förhållanden. Däremot riskerar den att störa användarna i utförandet av sina arbetsuppgifter.

Scenariobaserad utvärdering görs främst på produkter som håller på att tas fram. De utförs genom att utvalda testsubjekt ges uppgifter, eller scenarier, att slutföra. Subjekten studeras vid utförandet och ges en mängd frågor om sina upplevelser.

Gruppgranskning innefattar en expert i användbarhet som leder ett lag av främst användare men även ibland även exempelvis ergonomer. Utifrån den avsedde användarens arbetsuppgifter tas scenarier fram. Dessa diskuteras och utvärderas av gruppen och resultatet sammanställs. Framgångsfaktorer för metoden är att alla i gruppen hörs på lika villkor, att gruppen har en fungerande sammansättning samt att olika aktörer inte månar om sina egna intressen.

Laboratorieutvärdering I en laboratorieutvärdering placeras vanligtvis användaren i ett rum skilt från observatörerna. På så sätt tillåts dessa inte på något sätt påverka hur användaren löser sin uppgift. Utvärderingen görs först efter det att användaren slutfört sina uppgifter. Sessionen dokumenteras vanligtvis genom videoinspelning.

En nackdel med metoden är att användaren plockas ur sin vanliga miljö och in i en obekant. Vidare är det förenat med höga kostnader då särskilda lokaler och apparater krävs.

Även för testmetoder för användbarhet finns det en ISO-standard, ISO/TR 16982:2002 (Ergonomics of human-system interaction — Usability methods supporting human-centered design).

2.3 Loggning av användardata

Loggning av användardata består huvudsakligen av två steg, instrumentering av applikationen samt analys av den insamlade datan. De metoder som studerats i tidigare arbeten sammanställs nedan.

2.3.1 Instrumenteringsmetoder

Olika tilvägagångssätt för att instrumentera presenteras nedan.[2]

Manuellt Handpåläggningar där specifika komponenter i användargränssnittet uttryckligen väljs ut för loggning är en enkel metod för mindre och fokuserade mer tester. Däremot blir metoden snabbt ohållbar i takt med att testerna växer, både avseende tidsinsatsen för själva instrumenteringen och den införda komplexiteten i koden. Metoden är i sig påträngande för den studerade applikationen och kopplar instrumenteringen tätt till de berörda modulerna.

I ramverket Genom att istället införa instrumenteringen centralt i presentationsramverket utförs ändringarna på ett ställe. Samtidigt förskjuts ökningen av komplexitet i koden hit istället. Nackdelar med metoden är den potentiellt stora mängden händelser som riskerar att behöva registreras eller ens hanteras samt att relevant information om t.ex. sammanhang för händelsen högre upp i hierarkin inte nödvändigtvis är tillgänglig.

Interaktivt I sin undersökning skapade Bateman en metod som de döpte interaktiv användbarhetsinstrumentering (*eng. Interactive Usability Instrumentation*). Denna gav användbarhetsingenjören ett grafiskt verktyg för att välja ut de komponenter i applikationen som skulle studeras. Med hjälp av aspektorienterad[2] programmering vävdes instrumenteringen in i Java-

[2]Aspektorienterad programmering tillåter ökad modularisering genom att stödja separationen av tvärskärande angelägenheter. Vanliga programmeringsparadigm erbjuder vanligtvis

applikationens bytekod vid körtid. Fördelar med metoden är bl. a. att instrumenteringen inte kräver kodändring och att påverkan på applikationen hålls till ett minimum.

2.3.2 Analysmetoder

Ett urval av analysmetoder från annan forskning introduceras översiktligt nedan.[7]

Transformering av händelseströmmar syftar till att göra det lättare för människor och maskiner att behandla innehållet. Processen består av tre steg: urval, abstraktion och omkodning (*eng. selection, abstraction and recoding*).

I steget *urval* sållas bakgrundsbrus bort. Det kan ske genom att antingen positivt välja ut vissa händelser som intressanta, eller negativt välja bort andra som ointressanta. Här måste speciell omtanke tas så att inte urvalet förstör mätdatan. Steget kompliceras givetvis i de fall undersökaren valt att samla in all tänkbar information.

Steget *abstraktion* syftar till att från en mängd händelser på lägre nivå bilda nya på en högre nivå. Exempelvis kan händelserna då användaren först navigerar i menyn, därefter väljer alternativet skriv ut och sedan accepterar dialogrutan för utskrift istället förstås som att helt enkelt *skriva ut*. På detta sätt kan händelser på konceptuell nivå observeras och behandlas maskinellt.

Genom att *koda om* händelseströmmen skapas nya händelseströmmar utifrån de abstraherade händelserna i det föregående steget. Detta underlättar det fortsatta arbetet och bevarar rådatan i de gamla strömmarna.

Summering är en förhållandevis enkel metod för att svara på frågor om prestationer, frekvenser, byte mellan inmatningsenheter o.s.v. En fördel med metoden är att den kan ofta att utföras med hjälp av kalkylbladsprogram, databasfrågor eller generella data mining-verktyg.

Synkronisering och sökning är en metod där de observerande studierna kombineras med analys av loggade händelser. Syftet är att fånga information

möjligheter att modularisera genom klasser, paket, metoder osv. Vissa angelägenheter kan dock inte separeras på ett användbart sätt med hjälp av dessa metoder — de går över de nivåer av abstraktion som upprättats i programmet.

på en högre nivå som inte nödvändigtvis går att utläsa från händelseströmmen. Praktiskt sett kan detta utföras genom att användaren filmas eller bildskärmen spelas in för att sedan spelas upp tillsammans med den registrerade händelseströmmen. Undersökaren kan med den här metoden gå från en intressant händelse i inspelningen till den relevanta händelsen i strömmen, och vice versa.

Även om denna kombination syftar till att väga upp de båda metodernas för- och nackdelar återstår många av besvären med inspelningsutrustning, påverkan på testsubjekten och de förhållandevis stora mängderna information som kräver bearbetning. Metoden går inte heller att skala upp på ett praktiskt genomförbart sätt.

Visualisering sträcker sig från olika sätt att presentera fynd i form av grafer och diagram till att rita interaktioner ovanpå bilden av det studerade gränssnittet. Genom att visualisera datan kan den göras tillgänglig för andra än undersökarna. Det kan även hjälpa undersökarna genom att datan blir mer intiutiv.

2.4 Ett exempel från verkligheten

Inför omarbetningen av Microsofts Office-paket genomfördes ett projekt där användarna erbjöds bidra med sin egen användningsdata. Insamlingen var planlös och sträckte sig så långt som att samla allt som inte kränkte användarnas integritet. I väntan på resultatet gjorde utvecklarna förutsägelser om vilket som var det mest använda kommandot i ordbehandlingsprogrammet Word. Förutsägelserna motiverades med olika utsagor och anekdoter.

Vid analysen av datan fann man att de fem vanligaste kommandona utgjorde hela 32 %, därefter avtog användningsfrekvensen drastiskt. Det i särklass vanligaste kommandot var *klistra in* med 11 %, vilket var dubbelt så mycket som för det näst vanligaste kommandot *spara*. Detta ledde till att dessa kommandon kunde framhävas i det nya gränssnittet.

Vad som kanske var mest intressant med projektet var antagandet om att det skulle vara så pass få användare som trycker på knapparna för *kopiera* och *klistra in* att dessa likväl kunde tas bort från gränssnittet (vem använder inte kortkommandot på tangentbordet?). Tvärt emot detta användes de facto knapparna. Tack vare den insamlade datan kunde man alltså fatta ett bättre underrättat beslut jämfört med ett baserat på antaganden.[6, 5]

3 Metod

Utgångspunkten för val av metod i projektet är att söka svar på de tre huvudsakliga frågorna (se 1.2). Vidare ska metoden fortsätta att bygga på kunskaper från annan forskning på området, för att slutligen applicera dessa på ett sätt som passar in i organisationen. För att det slutförda arbetet senare ska kunna utvärderas inleds planeringen med en uttrycklig genomförandeplan och därtill hörande målsättning.

3.1 Planering, förarbete och målsättning

Målet med projektet är att färdigställa en prototyp för loggning av användardata i den studerade applikationen och utvärdera hur den insamlade datan kan analyseras.

Arbetet påbörjades med att planlägga vilka moment som utförs under den tilldelade tiden. De två första veckorna avsattes till att studera forskning på området i syfte att inte uppfinna hjulet två gånger. Baserat på de fynd som gjordes i litteraturstudien utformades resten av projektet.

Vecka	**Moment**
1-2	Kravspecifikation, litteraturstudie, avgöranden kring arkitektur
3-7	Utveckling av prototyp
8-9	Testning, utvärdering och rapportskrivning
10	Rapport, abstract, framsida och slutredovisning färdiga

Tabell 1: *Preliminär tidsplanering.*

3.2 Val av instrumentering

Den metod som väljs för att instrumentera applikationen bör uppfylla de krav eller riktlinjer som nämnts om minsta möjliga intrång i koden och möjligheten kunna genomföras på ett praktiskt och effektivt sätt.

Att manuellt införa en begränsad uppsättning handpåläggningar i koden för de intressanta interaktionspunkterna uppfyller inte behovet att kunna svara på frågor som vilka delar av gränssnittet används eller inte.

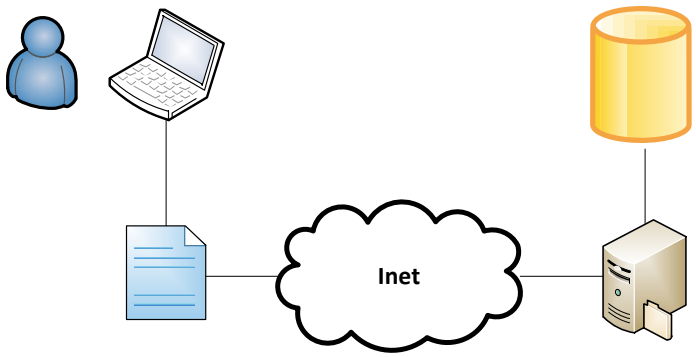
Tanken på att skapa ett grafiskt gränssnitt för att välja ut komponenter för registrering är i sig attraktiv, men görs dessvärre omöjlig av olika omständigheter. Bland annat skulle utvecklingen av själva verktyget gå förbi vad som

är genomförbart med hänsyn till de givna ramarna. Det skulle även lägga ett orimligt stort fokus på den ena av de ställda frågorna, närmre bestämt **Q1**.

Mot denna bakgrund och tillsammans med applikationens betydande storlek, framstår därför en ramverksinstrumentering som mest lämpad. Den ger möjlighet att både samla upp hela gränssnittet och samtidigt samla ökningen i komplexitet till ett ställe.

3.3 Prototyp

I syfte att påvisa ett genomförbart koncept och samtidigt dra lärdom av påstötta fallgropar framställs en fungerande prototyp. Denna omfattar stegen ända från registreringen av själva interaktionerna, tillfällig lagring av händelseströmmen på användarens dator till insamligen på en central plats för senare behandling.



Figur 2: *Konceptuell skiss över delarna av prototypen.*

Prototypen kan köras lokalt på en av arbetsgruppens byggdatorer. De "användare" som då träffas av instrumenteringen är de utvecklare som arbetar i den senaste utgåvan (2.13). Utöver utvecklarna kommer även automatiserade GUI-tester utföras på utgåvan och därmed generera ytterligare data. Det kan vara av intresse att senare kunna urskilja dessa sessioner från utvecklarnas. Då även metodingenjörerna får tillgång till senaste revisionen kommer också deras interaktioner att registreras och rapporteras.

3.4 Analysmetod

Analysen av resultatet syftar till att besvara de två frågorna utvecklingsorganisationen såg som mest intressanta och som samtidigt har en rimlig chans att

faktiskt bli besvarade.

I SDP3 kan reparatören inledningsvis välja mellan att utföra en uppsättning olika *jobbtyper*, så som kontroll och justering, ombyggnation, tillsyn m.m. En tanke som funnits inom organisationen har varit att optimera applikationen för den mest använda av dessa. Själva interaktionen användaren gör för detta är att trycka på en av de sex knapparna. Givet att varje knapptryckning ger upphov till en händelse i loggen torde ett svar finnas genom att summera dessa händelser.

Den andra analysfrågan gäller huruvida reparatören använder sig av en viss typ av presentationsvy. Då utvecklingen av denna vy tar i anspråk en betydande mängd trängda resurser, är det intressant att veta om det motiveras av funktionalitetens nytta och användning. Det har även varit på förslag att vyn borde vara förvald om den skulle den visar sig välanvänd.

Vad gäller den andra frågan kompliceras däremot analysen. Detta beror främst på att endast tre av de sex jobbtyperna erbjuder den aktuella vyn. Frågan betingas därmed av om någon av de intressanta jobbtyperna utfördes. Vidare bör det även vägas in i analysen det fall då användaren inom en kort tid byter tillbaka till det nuvarande förvalet.

4 Genomförande

4.1 Tidsplan och diskussioner

En viktig del av förberedelserna var att diskutera olika för- och nackdelar med de tänkbara tillvägagångssätten tillsammans med utvecklarna. Tidigt dök frågan om mätningarnas validitet och reliabilitet upp och har sedan funnits i åtanke under hela projektet. Många av diskussionerna rörde även tänkbara svar den insamlade datan kunde ge ledning i, exempelvis om det finns delar av applikationen som aldrig testats eller om användare i olika delar av världen skiljer sig i sina användningsmönster.

4.2 Studiebesök verkstad

För att få en ökad förståelse av hur reparatören använder SDP3 förankrades de insamlade kunskaperna med ett besök på demoverkstaden YSS inne på bolagets egna område.

Vid tillfället för besöket utförde en grupp reparatörer ett trycktest på en

tank för Urea[3]. En av reparatörerna hämtade efter slutförd montering en bärbar dator med SDP3 och anslöt den till lastbilen. Applikationen startades och reparatören navigerade sig snabbt till det rätta stället i applikationen. Efter detta fälldes helt enkelt datorn ihop och lades på hyllan varpå anslutningen kopplades ur.

4.3 Intervju funktionsägare

En längre intervju genomfördes med funktionsägaren för SDP3. Denne har en bakgrund som metodingenjör och därmed en god förståelse både för användarens behov och för metodingenjörerna.

Det som diskuterades var huvudsakligen vilka behov reparatören har samt om – och i så fall hur – dessa uppfylls av SDP3. Vad som framstår som den främsta frågan man söker svar på var den om användningen av användarfunktionsvyn.

Som förval presenteras reparatören med ett träd över elsystemet (se figur 1). Trädet är organiserat utifrån de olika komponenterna i elsystemet. Detta är i sig en logisk representation av hur själva hårdvaran är kopplad. Vad som däremot är problematiskt är exempelvis de fall en chaufför kommer in till verkstaden med problem på farthållaren. Eftersom detta är en distribuerad funktion så sköts den av fler än en styrenhet. Vyn över elsystemet ger ingen ledning i denna frågan om reparatören inte har kännedom om vilka servrar som sörjer för funkionen. Sådan kännedom anses som tämligen kvalificerad och är därför något verktyget bör ge ledning i.

För att bättre bistå reparatören har en vy skapats som grupperar trädet efter funktioner föraren nyttjar. Denna är dock inte den förvalda presentationsvyn då reparatören utför ett arbete utan måste väljas aktivt. Som nämnts är det förenat med besvär att utveckla vyn. Här uttrycks önskemålet om att veta i vilken usträckning reparatören ute i verkstaden faktiskt navigerar sig till den här vyn.

4.4 Prototypen

Det första steget i att upprätta prototypen var att framställa en kravspecifikation. I denna sammanställdes vilken data som bör registreras. Det framstod snart att för varje loggad session bör det i sin tur registreras start- och stoppti-

[3]Urea är ett tillsattsämne för minskning av partikelutsläpp.

der, användarkonto och vilken version av instrumenteringen som har använts (*specifikation*) samt vilken version av operativsystemet programmet körts i. För varje loggad händelse bör det registreras någon form av särskiljande ID, tidsstämpel och etikett eller annan beskrivande textrepresentation.

Den registerade informationen ska automatiskt lagras och sedan vid ett lämpligt tillfälle sändas över till en central plats. Datan ska vara ordnad på ett strukturerat sätt för att underlätta åtkomst av den, förslagsvis i en relationsdatabas.

4.4.1 Instrumenteringen

För att registrera interaktioner i WPF bör det noteras att ramverket använder något som heter routed events (*sv. dirigerade händelser*). Routed events har egenskapen att de kan låta varsko lyssnare i ett helt träd, inte nödvändigtvis enbart de objekt som utlöste dem från första början.[4] För att anmäla en lyssnare för givet ett routed event används klassen `EventManager`. Denna klass har en metod `RegisterClassHandler` till vilken en lyssnare (statisk metod) anmäls.

Instrumenteringen utfördes därefter genom att skapa en ny klass, `UsageLogger`. Denna fick en statisk bekvämlighetsmetod döpt `ObserveEvent` (se B.1). Metoden tar emot ett argument som ger uttryck för vilken typ av routed event som ska utlösa loggningsmekanismen och registrerar sedan sin egen lyssnare hos klassen `EventManager`.

Sedan tidigare körs automatiska GUI-tester på gränssnittet. För att testdatorn ska kunna anropa komponenter utan att klicka runt med muspekaren har varje komponent ett unikt ID, `Automation_Id`. Detta ID har utnyttjats för att kunna särskilja de unika komponenterna i gränsnittet så som det uttrycktes i kravspecifikationen.

4.4.2 Lagringen

För lagringen av händelseströmmen på användarens dator utnyttjades den redan befintliga infrastrukturen för loggning. Genom att helt enkelt införa ett nytt "mål" för loggning skrevs händelserna till en ny fil. Loggfilen inleds med en uppsättning rader som ger information om själva sessionen och miljön den körts i, därefter följer själva händelseströmmen. Varje händelse uttrycks som en rad med tidsstämpel, typ av händelse, i förekommande fall ett givet namn

[4] `http://msdn.microsoft.com/en-us/library/ms742806.aspx`

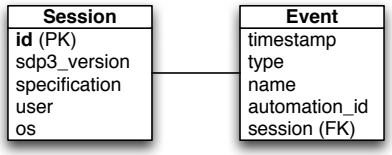
på komponenten samt komponentens automatiserings-ID. Fälten skiljs åt av semikolon. Skrivbufferten spolas för varje händelse och fångar därför även de sista händelserna innan en programkrasch. En rad kan se ut som nedan.

```
20120813 12:31:05:634 DEBUG Selector.Selected;;elErrorCodeTab
```

När användaren avslutar programmet är loggen komplett och redo att skickas in till centralen. Med hänsyn till att programslutet inte alltid är kontrollerat skickar programmet istället in loggar vid starten av programmet. En session rapporteras alltså först när en ny påbörjas.

4.4.3 Översändningen

För att kunna samla användningsdatan på en central plats upprättades en webbtjänst på arbetsgruppens byggdator Jenkins. Genom att utnyttja de möjligheter som Windows Communication Foundation (WCF) erbjuder kunde webbtjänsten snabbt färdiställas. Tjänsten tar emot loggfiler som den sedan lagrar i en relationsdatabas. Klockstämplar i loggfilen tolkas och görs om till korrekt datatyp i databasen.



Figur 3: *Datamodellen för relationsdatabasen.*

För varje fil som skickas in skapas en ny `Session`. Varje händelse i den filen motsvarar en post i tabellen `Event`. Ur dessa tabeller togs även en SQL-vy fram som uttrycker sessionen med tidsstämplar för start och stopp. Detta underlättar den senare behandlingen av datan.

4.4.4 Visualiseringen

För att tillgängliggöra datan för en bredare publik upprättades även en webbsida som presenterar den senaste datan i diagram och tabeller. Presentationen visar

- antalet sessioner,
- antalet sessioner per vecka de senaste tio veckorna samt

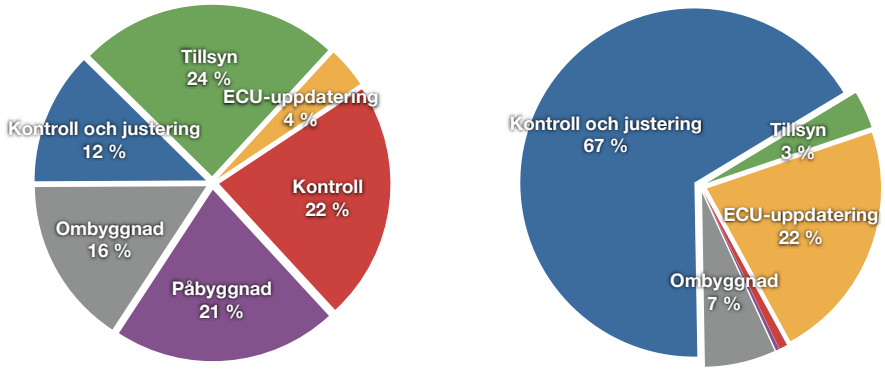
- minsta, genomsnittliga och maximala längderna för sessioner

Utöver dessa värden presenteras även fördelningen mellan de olika jobbtyperna i ett cirkeldiagram, dvs. resultatet för den första frågan.

5 Resultat

Här redovisas resultaten från den data som samlats in under projektets gång. Det bör dock noteras att samtidigt detta är intressanta observationer är det metoden och genomförbarheten som är av vikt i projektet.

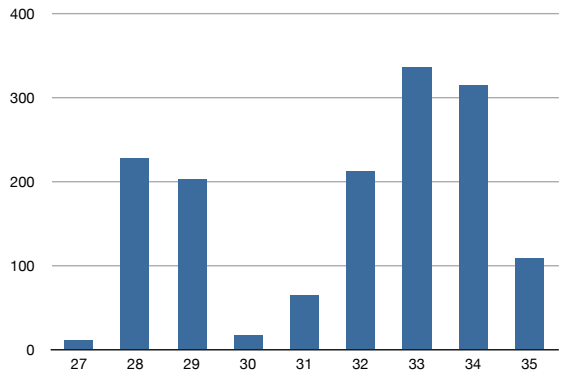
Under de nio veckor prototypen samlade in användardata uppgick antalet sessioner till 1 595 bestående av 25 941 händelser. Av dessa var 52 % av sessionerna genererade av GUI-testerna.



Figur 4: *Vänster: fördelningen av jobbtyper från GUI-testerna. Höger: fördelningen av jobbtyper från utvecklarnas arbetsstationer.*

Jobbtyp	**Användare**	**GUI-tester**
Kontroll och justering	425	81
ECU-uppdatering	142	25
Ombyggnad	42	102
Tillsyn	22	159
Kontroll	5	146
Påbyggnad	2	137

Tabell 2: *Antalet förekomster av de olika jobbtyperna uppdelat på de från GUI-testerna och utvecklarna.*



Figur 5: *Fördelningen av rapporterade sessioner per vecka.*

6 Slutsatser

Det resultat som presenteras i 5 måste inte bara ställas mot varandra utan även mot det som förväntades inledningsvis. Eftersom de automatiserade GUI-testerna är fördefinierade i skript och avsedda att säkerställa funktionaliteten i gränssnittet torde fördelningen mellan de olika jobbtyperna vara jämn. Detta matchas även av de fynd som gjordes vid insamlingen. På liknande sätt förväntades det att jobbtypen *kontroll och justering* skulle vara den mest förekommande bland utvecklarna eftersom knappen är placerad längst upp ligger nära till hands vid en snabbkörning.

Alltså går det att sluta sig vid att den redovisade metoden (se bilaga A för SQL-fårgorna) räcker för att svara på en fråga som den rörande vanligaste jobbtypen.

Den andra frågan, angående vilken utsträckning användare utnyttjar *användarfunktionsvyn*, visade sig vara mer omständig att besvara. Se diskussionen för vidare ledning.

7 Diskussion

Det kan här diskuteras varför de utnyttjade SQL-frågorna räckte till för att svara på frågan angående jobbtyper, medan de inte räckte till för att svara på frågan om användarfunktionsvyn. Detta beror främst på skillnaden mellan frågorna och hur detta representeras i datan. Vad gäller jobbtyper är det allt som oftast den första åtgärden användaren gör i gränssnittet. Alltså är händelsen *välja jobbtyp* inte betingad av andra händelser. Finns händelsen i loggen

är det tillräckligt säkert att jobbet har utförts för att det ska vara värt att räkna med det. Vidare är det tacksamt att räkna komplementet till en jobbtyp som summan av alla de andra observerade jobbtyperna då listan är uteslutande.

Gällande frågan i vilken utsträckning användarfunktionsvyn används. Notera först att denna vy endast är tillgänglig i tre av de sex jobbtyperna. Alltså är frågan betingad vilken jobbtyp som väljs. Här vore en tänkbar väg att subtrahera bort antalet sessioner som inte hade någon förekomst av dessa jobbtyper i sin händelseström. Dessväre kompliceras frågan vidare av möjligheten att användaren väljer vyn för att sedan byta tillbaka eller väljer bland fler flikar och vyer. Det måste således finnas ett sätt att med kod avgöra vad som borde utgöra att *användaren faktiskt utnyttjar vyn*. Av uppenbara skäl låter sig SQL-frågor inte göra detta på ett sätt som krävts genom projektet, nämligen på ett rationellt sätt som ger effektiva resultat.

Vad gäller ett skarpt genomförande av ett dylikt projekt finns det för närvarande en huvudsaklig utmaning. Att över internet ta emot loggfilerna kräver publik exponering av en resurs. Sådant görs redan i form av insänd driftdata från lastbilarna via SDP3. Öppnandet av en ny tjänst uppskattas ta i storleksordning 6 månader att slutföra. Vad som däremot talar för ett sådant genomförande är dels att den informationen som sänds över är klart *mindre* känslig, dels att nytta kan dras av att se till lösningen för översändning av driftdatan.

8 Rekommendationer

Utifrån de fynd som gjorts under projektets gång står det klart att bolaget har en möjlighet att med förhållandevis liten insats åstadkomma en skarp instrumentering och tillhörande tjänst för datainsamling. En sådan insamling kan så som i fallet med Microsoft ske planlöst för att sedan utföras när tillfälle ges. Det skulle då vara av betydelse att sessioner utmärker vilken version av programmet och *vad* som är instrumenterat (kallat *specifikationen* i projektet). Vill man i ett senare läge analysera datan är det av vikt att inte blanda ihop olika utföranden av gränssnittet.

Med tanke på lättheten för en utvecklare att i dagsläget testköra frågor från SQL-konsollen bör man överväga vidare vilka ytterligare frågor som kan tänkas lämpliga för detta. Framgångsfaktorer är då att det är enkelt att härleda en företeelse till en atomisk händelse i strömmen. Vidare bör även företeelsens komplement vara lättillgängligt.

Vid mer avancerade analyser bör det övervägas om det är mest fördelaktigt att antingen ta fram ett eget verktyg för att analysera loggarna utifrån vissa uppbyggda grammatiker, eller utvärdera de finns kommerciella verktyg på marknaden för en sådan uppgift.

A SQL-frågor

```
1  -- Convenience view
2  CREATE VIEW session_with_timestamps AS
3  SELECT s.id, s.sdp3_version, s.specification, [user], os,
4            start =
5            (SELECT min(timestamp)
6             FROM dbo.Event WHERE session = s.id),
7            finish =
8            (SELECT max(timestamp)
9             FROM dbo.Event WHERE session = s.id)
10 FROM dbo.Session s;
11
12 -- Summarize number of sessions for the last 10 weeks
13 SELECT TOP 10 week_no = DATEPART(WEEK, start), [count] = COUNT(*)
14        FROM session_with_timestamps
15        WHERE start IS NOT NULL
16        GROUP BY DATEPART(WEEK, start)
17        ORDER BY week_no;
18
19 -- Summarize number of sessions
20 SELECT COUNT(*) FROM dbo.Session;
21
22 -- Summarize minimal, average and maximal duration time
23 -- for sessions
24 SELECT duration = MIN((DATEDIFF(second, start, finish)))
25     FROM session_with_timestamps s
26     WHERE start IS NOT NULL AND finish IS NOT NULL;
27
28 SELECT duration = AVG((DATEDIFF(second, start, finish)))
29     FROM session_with_timestamps s
30     WHERE start IS NOT NULL AND finish IS NOT NULL;
31
32 SELECT duration = MAX((DATEDIFF(second, start, finish)))
33     FROM session_with_timestamps s
```

```
34      WHERE start IS NOT NULL AND finish IS NOT NULL;
35
36
37  -- Summarize number of job types performed
38  SELECT COUNT(*)
39      FROM dbo.Event
40      WHERE automation_id = 'ShowAllJobTypes.Button.RepairSCANIA';
41
42  SELECT COUNT(*)
43      FROM dbo.Event
44      WHERE automation_id = 'ShowAllJobTypes.Button.Maintain';
45
46  SELECT COUNT(*)
47      FROM dbo.Event
48      WHERE automation_id = 'ShowAllJobTypes.Button.Campaign';
49
50  SELECT COUNT(*)
51      FROM dbo.Event
52      WHERE automation_id = 'ShowAllJobTypes.Button.Check';
53
54  SELECT COUNT(*)
55      FROM dbo.Event
```

B Instrumentering

B.1 Verktyg

```
/// <summary>
/// Make the logger listen to the specified event.
/// </summary>
/// <param name="routedEvent"></param>
public static void ObserveEvent(RoutedEvent routedEvent)
{
        EventManager.RegisterClassHandler(typeof(Window),
                routedEvent,
                new RoutedEventHandler(UserEventHandler),
                true);
}

/// <summary>
/// Handles events comming in from the EventHandler.
/// </summary>
/// <param name="sender"></param>
/// <param name="args"></param>
private static void UserEventHandler(object sender, RoutedEventArgs args)
{
        FrameworkElement src = args.OriginalSource as FrameworkElement;

        if (src != null)
        {
                Log.WriteDeveloperLogEvent(LogTarget.UsageLogger,
                        String.Format("{0};{1};{2}",
                        args.RoutedEvent,
                        AutomationProperties.GetAutomationId(src),
                        src.Name));
        }
}
```

B.2 Urval

```
// Gather environment information
Dictionary<string, string> env = new Dictionary<string, string>();
env.Add("User", Environment.UserName);
env.Add("SDP3Version", string.Format("{0}.{1}.{2}.{3}",
        SDP3AppSettings.Setting("MajorVersion"),
        SDP3AppSettings.Setting("MinorVersion"),
        SDP3AppSettings.Setting("Revision"),
        SDP3AppSettings.Setting("Build")));
env.Add("OS", Environment.OSVersion.ToString());
env.Add("Specification", "1");

// Choose event types to be logged
UsageLogger.ObserveEvent(Button.ClickEvent);
UsageLogger.ObserveEvent(MenuItem.ClickEvent);
UsageLogger.ObserveEvent(System.Windows.Contr[..]s.Selector.SelectedEvent);
UsageLogger.ObserveEvent(TreeViewItem.SelectedEvent);
UsageLogger.ObserveEvent(TreeViewItem.ExpandedEvent);

// Initialize the logging utility
Scania.Sdp.UsageLogger.Setup(env);
```

Referenser

[1] *Annual report*. Scania, 2011.

[2] Scott Bateman, Carl Gutwin, Nathaniel Osgood, and Gordon I. McCalla. Interactive usability instrumentation. In T. C. Nicholas Graham, Gaëlle Calvary, and Philip D. Gray, editors, *EICS*, pages 45–54. ACM, 2009.

[3] A. Girgensohn, D.F. Redmiles, and F.M. Shipman III. Agent-based support for communication between developers and users in software design. In *Knowledge-Based Software Engineering Conference, 1994. Proceedings., Ninth*, pages 22–29. IEEE, 1994.

[4] Jan Gulliksen and Bengt Göransson. *Användarcentrerad systemdesign : en process med fokus på användare och användbarhet*. Studentlitteratur, Lund, 2002.

[5] Jensen Harris. Inside deep thought (why the ui, part 6), april 2006.

[6] Jensen Harris. No distaste for paste (why the ui, part 7), april 2006.

[7] David M. Hilbert and David F. Redmiles. Extracting usability information from user interface events. *ACM Comput. Surv.*, 32(4):384–421, December 2000.

[8] Jonathan Lazar, Jinjuan Heidi Feng, and Harry Hochheiser. *Research methods in human-computer interaction*. John Wiley, Chichester, 2010.

[9] Torsten Thurén. *Vetenskapsteori för nybörjare*. Liber, Stockholm, 2., [omarb.] uppl. edition, 2007.