Eero Laukkanen

Tiedonkeruu ohjelmointivirheiden raportoinnissa

Elektroniikan, tietoliikenteen ja automaation tiedekunta

Kandidaatintyö
Espoo 31.8.2010

Vastuuopettaja:

TkT Pekka Forsman

Työn ohjaaja:

TkT Mika Mäntylä

Aalto-yliopisto
Teknillinen korkeakoulu

AALTO-YLIOPISTO
TEKNILLINEN KORKEAKOULU

KANDIDAATINTYÖN
TIIVISTELMÄ

Tekijä: Eero Laukkanen

Työn nimi: Tiedonkeruu ohjelmointivirheiden raportoinnissa

Päivämäärä: 31.8.2010 Kieli: Suomi Sivumäärä:6+34

Tutkinto-ohjelma: Automaatio- ja systeemitekniikka

Vastuuopettaja: TkT Pekka Forsman

Ohjaaja: TkT Mika Mäntylä

Ohjelmointivirheiden raportointi on olennainen osa tietokoneohjelmiston elinkaarta, mutta tutkimuksissa on havaittu, että ohjelmistokehittäjien saamat virheraportit ovat usein puutteellisia ohjelmointivirheiden korjaukseen. Virheraportoinnissa raportoijalta vaaditaan hyvää teknistä tietämystä, jota ohjelmiston jokaisella käyttäjällä tai testaajalla ei ole.

Tässä työssä selvitettiin mahdollisuuksia helpottaa virheraportoijan työtä automatisoimalla virheraportoinnin tiedonkeruuta. Ohjelmistokehittäjille tehdyllä Internet-kyselyllä tutkittiin, mitkä virhetiedot ovat hyödyllisiä ohjelmointivirheiden korjauksessa, kartoitettiin eri ohjelmisto-organisaatioiden virheraporttien laatu ja kysyttiin kehittäjien mielipiteitä tiedonkeruun automaation toteutusmahdollisuuksista.

Vastausten perusteella kehittäjille hyödyllisimpiä virhetietoja ovat ohjeet virheen toistamiseen ja sovelluksen osa, jossa virhe tapahtui. Usean muun virhetiedon hyödyllisyys riippuu virheestä tai ohjelmiston suoritusympäristöstä. Hyödyllisissä virhetiedoissa on esiintynyt puutteellisuutta tai virheellisyyttä virheraporteissa ohjelmisto-organisaatiosta riippumatta. Virheraportoinnissa yleisluontoisten tietojen kuten virheen toisto-ohjeiden keräämistä voidaan helpottaa huomattavasti esimerkiksi nauhoittamalla ohjelmiston käyttöä. Yksityiskohtaisempien tietojen kuten suorituspinon kerääminen vaatii, että kerättävät virhetiedot on määritelty ohjelmistokohtaisesti. Virheraportoinnin toteutuksessa tulee ottaa huomioon erot ohjelmiston loppukäyttäjän ja testaajan välillä.

Avainsanat: tiedonkeruu, bugi, ohjelmointivirhe, virhetiedot, ohjelmistotestaus, virheraportointi, virheraportointijärjestelmä, virheenseuraamisjärjestelmä

Esipuhe

Tämä kandidaatintyö tehtiin kesällä 2010 Aalto-yliopiston ohjelmistotuotannon ja -liiketoiminnan laboratorion ESPA-tutkimusprojektissa. Työn tekemisen aikana kävin monia opettavaisia keskusteluja kandidaatintyön kirjoittamisesta tutkimusprojektin tutkijoiden kanssa. Timo Lehtinen antoi rakentavia kommentteja kyselyn muodostamisesta ja tekstin rakenteesta. Jari Vanhanen antoi samankaltaisesti asiantuntevia neuvoja kyselyn muodostamisessa. Mika Mäntylä mahdollisti työn ohjaajana riittävän vapauden työn tekemiselle ja oli myös asiantuntevana tukena työtä suunniteltaessa. Häneltä sain erityisesti työn rakenteen kannalta arvokkaita kommentteja. Suuri kiitos kuuluu näille henkilöille työn valmiiksi saattamisessa.

Tutkimusaineisto on tutkimuksen raaka-aine. Vaikka metodit olisivat täydellisiä, tuloksiin vaaditaan myös aineistoa. Ilman tutkimusprojektin yhteistyöyrityksiä ja yhteyshenkilöitä työn kaltaista aineistoa ei olisi saatu kerättyä. Yhteyshenkilöiden lisäksi kyselyn vastaajien eli yritysten ohjelmistokehittäjien työpanos oli kiitoksen arvoinen.

En tehnyt työtä itsenäisesti omalla ajallani, vaan kävin työhön liittyen monia ajatuksia herättäviä keskusteluja minulle läheisten ihmisten kanssa. Myös nämä keskustelut olivat tarpeellisia – ne toimivat henkireikinä aikoina, jolloin edistystä ei tuntunut tapahtuvan. Kiitokset kaikille niille nimeämättömille, jotka keskustelivat kanssani mm. LaTeX:n ihmeellisestä maailmasta.

Otaniemi, 31.8.2010

Eero I. Laukkanen

Sisältö

Määritelmät

Avoimen lähdekoodin ohjelmistokehitys käsittää kaiken ohjelmistokehityksen, jossa ohjelmiston lähdekoodi on kaikkien saatavilla. Vastakohtana on suljettu ohjelmistokehitys.

Manuaalinen virheraportointi tarkoittaa käyttäjän aloittamaa virheraportointia. Vastakohtana on automaattinen virheraportointi, jossa virheraportointi alkaa ohjelmiston keskeytyksen takia.

Muistivedos (engl. *core file, core dump*) sisältää työmuistin sisällön ohjelmiston suoritushetkellä. Käytännössä vedostustiedosto sisältää ohjelmiston tilan esimerkiksi ohjelmiston kaatuessa.

Ohjelmiston loppukäyttäjä käyttää sovellusta omiin tarkoitusperiinsä sovelluksen kehityksestä riippumatta.

Ohjelmiston suoritusympäristö on viitekehys, jossa ohjelmisto suoritetaan. Esimerkiksi Internet-sivustolla toimivan ohjelmiston suoritusympäristö eroaa sulautettujen järjestelmien suoritusympäristöstä.

Ohjelmointivirheen takia tietokoneohjelmiston suoritus poikkeaa ohjelmiston käyttäjän odotuksista.

Suorituspino (engl. *stack trace*) sisältää tiedon, missä kohtaa ohjelmakoodia sovellus on tietyllä ajanhetkellä.

Testaaja on henkilö, joka etsii virheitä ohjelmistosta sen laadun varmistamiseksi ja parantamiseksi.

Vianseurantajärjestelmän (engl. *bug tracking system*) avulla voidaan raportoida, seurata ja hallita ohjelmointivirheitä niiden koko elinkaaren ajan. Esimerkiksi Bugzilla [10] on vianseurantajärjestelmä.

Virhetiedot ovat ohjelmointivirheeseen liittyviä tietoja.

1 Johdanto

Ohjelmistotuotannossa väitetään olevan ongelmia ohjelmointivirheiden raportoinnissa virheiden korjaajille eli ohjelmistokehittäjille [8]. Virheiden korjaukseen tarkoitetut virhetiedot on varastoitu virhetietokantoihin, joita tarkasteltaessa virhetiedoissa on havaittu puutteita [8, 24]. Manuaalisessa virheraportoinnissa yksityiskohtaisten virhetietojen kerääminen lisää virheraportoinnin työmäärää, mikä voi aiheuttaa joidenkin virhetietojen syöttämisen ohittamisen. Virheitä raportoivat monet sidosryhmät, kuten ohjelmistotuotteiden käyttäjät, ohjelmistoyritysten testaajat ja markkinoijat [24], minkä takia raportoijien taustat, ohjelmistotekninen tietämys ja osaaminen vaihtelevat. Raportoijalta ei välttämättä löydy edellytyksiä riittävän hyvälaatuiseen manuaaliseen virheraportointiin. Virheraporteissa ilmenevien ongelmien takia kehittäjät joutuvat selvittämään puuttuvia virhetietoja raportoijalta tai itse tutkimalla, mihin tuhlaantuu tuotteliasta työaikaa. LaTozan ym. [21] mukaan suurin osa ohjelmistokehittäjän työajasta kuluu virheiden korjaukseen.

Ongelman ratkaisemiseksi on esitetty mm. virheraportoijien kouluttamista ja virhetietojen keräämisen helpottamista jonkinlaisen työkalun avulla [39]. Tässä työssä lähtökohtana on työkalun kehittäminen, koska edistyneellä virheraportointityökalulla myös kokemattomammat raportoijat pystyisivät raportoimaan virheitä paremmin. Tämän työn tavoitteena on siis selvittää mahdollisuudet virheraportoinnin kehittämiseen tiedonkeruun automatisoinnin ja helpottamisen osalta. Tavoitteet kiteytyvät työn tutkimuskysymyksiin:

1. *Mitkä virhetiedot ovat ohjelmistokehittäjille hyödyllisiä ohjelmointivirheiden korjauksessa?* Aikaisempien tutkimusten perusteella jotkin tiedot ovat selvästi tärkeitä jokaiselle kehittäjälle, kun taas joidenkin tietojen tarpeellisuudessa esiintyy enemmän hajontaa kehittäjien välillä [8]. Tässä työssä hypoteesina onkin, että hyödylliset virhetiedot eroaisivat virheellisen ohjelmiston suoritusympäristöstä riippuen.

2. *Missä ohjelmistokehittäjille hyödyllisissä virhetiedoissa on ollut puutteita tai virheitä virheraporteissa?* Aikaisempien tutkimusten mukaan virheraportoinnissa on ongelmia. Tutkimuksissa on todettu, että virheraporteissa esiintyy virheitä [8, 24] ja virheellisten raporttien kriittisyyttä on arvioitu [8]. Tässä työssä ongelman tutkimusta laajennetaan tutkimalla, missä virhetiedoissa ongelmia esiintyy.

3. *Miten kehittäjille hyödyllisiä virhetietoja voidaan kerätä virheraportteihin mahdollisimman vaivattomasti?* Aikaisemmat tutkimukset ovat ehdottaneet virheraporttien laadun parantamiseksi vianseurantajärjestelmiin työkalua, jolla raportoija voi kerätä tietoa virheraportteihin automaattisesti tai tietokoneavusteisesti [19, 24]. Tässä työssä pyritään määrittelemään mitä tietoja työkalulla olisi mahdollista ja kannattavaa kerätä.

Tutkimuskysymyksiin vastattiin eksploratiivisen tutkimuksen keinoin [20] uutta tietoa kartoittavalla kyselytutkimuksella. Työssä tehtiin puoliavoin kysely [14, s. 7–8] viiden eri yrityksen ohjelmistokehittäjille. Yritykset olivat tutkimusyhteistyössä tutkimusprojektin kanssa. Pohjana kyselylle oli aikaisempi Bettenburgin ym. tutkimuksessa tehty kysely [8], jota muokattiin vastaamaan tämän työn tavoitteita. Kyselyn vastausten perusteella pyrittiin vastaamaan sekä kvantitatiivisesti että kvalitatiivisesti työn tutkimuskysymyksiin.

2 Teoreettinen tausta ja aikaisempi tutkimus

Virheraportointiin liittyvä tutkimuskenttä on laaja ja sen rajaaminen on mahdollista tehdä monella eri tavalla. Tähän työhön kytkeytyvät erityisesti erilaiset virheraportointikäytännöt, virheraporttien laatu, kehittäjien tietotarpeet virheiden korjaukseen ja raportointijärjestelmien parantamiseen liittyvät tutkimukset. Luvun alussa kerrotaan, miksi virheraportointia ylipäänsä tarvitaan. Osissa 2.1 ja 2.2 esitellään tälle työlle erityisesti motivaatiota antaneet Mäntylän ym. [24] ja Bettenburgin ym. [8] tutkimukset.

2.1 Ohjelmistokehitys ja -testaus

Ohjelmistokehitys on tietokoneohjelmistojen tuottamista ihmisten erilaisiin tarpeisiin. Jacobson ja Bylund [17, s. 32] kuvaavat ohjelmistokehitysprosessia seuraavana kolmen vaiheen kokonaisuutena:

Analyysivaiheessa määritellään, miten ohjelmiston tulisi toimia ja mitä se tarjoaa käyttäjilleen. Samalla ohjelmistolle asetetaan yksiselitteiset vaatimukset. Ohjelmistolle määritellään selkeä ja kokonaisvaltainen rakenne, mutta käytännön toteutukseen yksityiskohtiin, esimerkiksi ohjelmointikieleen ja sen rajoitteisiin, ei puututa.

Suunnittelu- ja toteutusvaiheessa realisoidaan analyysivaiheessa muodostettu malli. Ensin suunnitellaan, mitä toteutusympäristöä käytetään ohjelmiston toteuttamiseen, minkä jälkeen suunnitelman perusteella toteutetaan muodostettava ohjelmisto.

Testausvaiheessa varmistetaan, että toteutusvaiheen lopputulos täyttää ohjelmistolle analyysivaiheessa asetetut vaatimukset.

Kehitysprosessi ei ole kyseisellä mallilla yksiselitteinen, vaan käytännössä mallin vaiheet toteutetaan eri tavoilla esimerkiksi ketterän ja suunnitelmaohjautuvan ohjelmistokehityksen [9] välillä. Jokainen vaihe on kuitenkin tarpeellinen menestyksekkään ohjelmistokehityksen kannalta. Sovelluksen koon kasvaessa se monimutkaistuu ja laadun varmistaminen on haasteellisempaa [33]. Testaamalla ohjelmistoa ennen sen julkaisua voidaan vähentää virheiden määrää merkittävästi.

Ohjelmistotestaus toteutetaan yleensä 3–4 eri tasolla tai vaiheella, jotka voidaan luokitella seuraavin termein [11, s. 133–135]:

Yksikkötestauksella tarkoitetaan yksittäisten ohjelmakomponenttien testausta. Ohjelmakomponentit ovat ohjelman osia, jotka muodostavat itsenäisesti jonkin pienemmän kokonaisuuden ohjelmassa.

Integraatiotestauksessa testataan yksikkötestauksessa käytettyjen yksiköiden välisiä rajapintoja ja yksiköiden yhteistoimintaa.

Systeemitestauksessa testataan ohjelmaa kokonaisuudessaan.

Hyväksymistestaus vastaa toiminnallisuudeltaan systeemitestausta, mutta samalla varmistetaan, että vaatimusmäärittelyssä tehdyt vaatimukset täyttyvät ohjelmiston suhteen.

Vaikka Jacobsonin ja Bylundin mallissa testaus on eritelty omaksi vaiheekseen ohjelmistokehitysprosessin lopussa, laadukkaan ja aikataulussa ja budjetissa pysyvän ohjelmiston tuottamista varten testaaminen on aloitettava niin aikaisin kuin on mahdollista, mikä toteutuu *V-mallissa* [11, s. 253–257]. Mallissa esitetään, kuinka eri testaustasojen testit kytkeytyvät analyysiin, suunnitteluun ja toteutukseen. Testit pystytään määrittelemään ja suunnittelemaan tai jopa toteuttamaan ennen kuin itse ohjelmisto on toteutettu.

Ohjelmistoa voidaan testata joko automaattisesti tai manuaalisesti. Automaattisen testauksen etuina olisivat testien helppo toistettavuus ja havaittujen virheiden yksiselitteinen raportointi, mutta automatisointi ei ole aina tilanteen mukaan mahdollista [6, 7, 25] ja yrityksissä sitä käytetään hyvin vähän [2]. Toisaalta manuaalisessa testauksessa ilmenee tarve virheraportoinnille.

Mäntylä ym. [24] tutkivat kolmen eri ohjelmistoyrityksen ohjelmistotestauskäytäntöjä. Kaikissa yrityksissä ohjelmistotestaus suoritettiin usean eri työntekijäryhmän voimin, ja esimerkiksi myynnin ja konsultoinnin parissa työskennelleet löysivät merkittävän osan yritysten ohjelmistojen ohjelmointivirheistä. Testaukseen ei käytetty pelkästään siihen erikoistuineita ammattilaisia, koska lähempänä asiakasta työskentelevät tuntevat asiakkaan työkentän ja näkökulman parhaiten. Testauskäytäntöjen kvantitatiivinen analyysi tehtiin analysoimalla yritysten virhetietokantoja, jolloin havaittiin virheraporttien sisältävän tietoa vaihtelevasti. Ratkaisuksi ongelmaan ehdotettiin virhetietojen automaattista keräämistä virheraportteihin.

2.2 Virheraportointi

Virheraportointia tarvitaan, kun ohjelmointivirheen löytää eri henkilö kuin virheen korjaaja. Virheraportointi toteutetaan usein vianseurantajärjestelmässä [11, s. 485–486], jossa ohjelmiston kehityksestä kiinnostuneet osapuolet voivat raportoida, seurata ja hallita virheitä virheen koko elinkaaren ajan. Vianseurantajärjestelmä on olennainen osa ohjelmistotestausta hallinnan ja valvonnan kannalta ja oikein käytettynä se parantaa ohjelmiston laatua ja vähentää tuotantokustannuksia [35]. Virheitä voivat raportoida esimerkiksi ohjelmiston loppukäyttäjät tai testaajat. Raportin kulkeutuminen kehittäjille vaihtelee järjestelmästä riippuen: avoimen lähdekoodin ohjelmistojen keskuudessa suosittuja ovat avoimet järjestelmät [4], joissa kuka tahansa voi raportoida virheitä suoraan kehittäjille. Yrityksissä asiakkaiden virheraportit voidaan käsitellä yrityksen tukihenkilöiden toimesta ennen järjestelmään lisäämistä [1].

Anvik ym. [4] tutkivat kahden avoimen lähdekoodin, Eclipsen ja Firefoxin, virheraportointijärjestelmien käytäntöjä ja ongelmia. Molemmissa järjestelmissä käytettiin Bugzilla-ohjelmistoa [10], jossa raportti muodostetaan raportointilomakkeessa käsinsyötettävillä tekstikentillä. Avoimissa järjestelmissä ongelmaksi huomattiin virheraporttien suuri määrä, joka aiheuttaa ylimääräistä tiedonhallinnallista työtä. Ongelmaksi koettiin myös raporttien hyöty ohjelmistoprojektille: raporttien luokittelun perusteella 39% Eclipsen ja 56% Firefoxin raporteista eivät auttaneet parantamaan projektia. Ongelmat vähenisivät, jos duplikaatit raportit ja raporttien jakaminen kehittäjille tehtäisiin automaattisesti. Erilaisia koneoppimismenetelmiä on tutkittu soveltuvan tehtävään jossain määrin [13].

Koska virheraporttien laadun on todettu olevan heikkoa [8, 24], on raportointi- ja seurantajärjestelmiin esitetty tutkimuksissa parannuksia. Zimmermann ym. [39] esittävät pa-

rannuksia, joiden tarkoituksena on lisätä virheraporttien kattavuutta virheiden korjaukseen. Tutkimuksessa esitetään neljä tapaa parantaa vianseurantajärjestelmiä:

1. Kehittämällä työkaluja, joilla käyttäjä voi kerätä tietoa automaattisesti.
2. Tarkkailemalla käyttäjän syöttämää tietoa automaattisesti ja parantamalla siten raporttien laatua.
3. Parantamalla virheenkorjausprosessia, kuten virheiden lajittelua korjaajille ja virheen tilan seurantaa.
4. Käyttäjiä kouluttamalla raportoijat tietävät virheidenkorjaukseen tarvittavat tiedot ja kuinka ne kerätään. Kehittäjiä voidaan kouluttaa käyttämään virheraporteista saatavaa tietoa paremmin.

Tutkimuksen lopussa esitetään koneoppimismenetelmiä käyttävä parannus vianseurantajärjestelmiin. Menetelmän avulla raportoija saa automaattisesti syventäviä kysymyksiä, joihin saatujen vastausten avulla voidaan ratkaista virheen sijainti ohjelmakoodissa.

Mitä virhetietoja virheraportteihin tulisi sisällyttää? Tietotarpeet virheiden korjaukseen riippuvat pitkälti virheen ominaisuuksista, ja riittävien tiedontarpeiden arviointi on vaikeaa [27]. Siksi on tärkeää kerätä niin paljon informaatiota kuin mahdollista. Kerättävät virhetiedot eroavat toisistaan riippuen siitä, voidaanko virhetieto kerätä automaattisesti vai tarvitaanko keräämiseen raportoijan vuorovaikutusta (Taulukko 1).

Taulukko 1: Virheraporttiin sisällytettävät virhetiedot eri lähteissä.

Virhetiedot	Bettenburg ym. [8]	Goldberg [15]	GNOME Bug Buddy [16]	Murphy [27]	Alnefelt ja Malmgren [1]
Manuaalisesti kerättävät					
virheen toisto-ohjeet	✓	✓	✓		
havaittu käyttäytyminen	✓	✓	✓		
odotettu käyttäytyminen	✓	✓	✓		
lisätiedot raportoijalta		✓	✓	✓	
Automaattisesti kerättävät					
suorituspinot	✓	✓	✓		✓
testitapaukset [5]	✓				
kuvakaappaukset	✓	✓			✓
muistivedokset			✓	✓	✓
virhehistoria			✓	✓	
tuotetiedot		✓	✓	✓	✓
muut käytetyt ohjelmistot				✓	✓
käyttäjän vuorovaikutus					✓

Bettenburg ym. [8] tekivät kyselyn kolmelle eri avoimen lähdekoodin ohjelmistoprojektin ohjelmistokehittäjille ja virheraportoijille eri virhetietojen tärkeydestä (Taulukko 1) ja ongelmista virheraportoinnissa. Kyselyn perusteella selvisi, että raportoijien lähettämien tietojen ja kehittäjien tärkeimmiksi koettujen tietojen välillä vallitsee ristiriita: raportoijat eivät lähetä niitä tietoja, jotka kehittäjät kokevat tärkeimmiksi. Raportoijat tietävät, mitä tietoja kehittäjät tarvitsevat, mutta joidenkin tietojen selvittäminen on vaikeaa. Toisaalta virheelliset ja puutteelliset tiedot aiheuttivat eniten ongelmia kehittäjille vir-

heiden korjauksessa. Kyselyn perusteella tehtiin CUEZILLA-työkalu, joka pystyi arvioimaan virheraporttien laatua ja ehdottamaan parannuksia niihin. Työkalu voisi olla raportoijan apuna raporttia täytettäessä.

Goldberg [15] esittelee hyvän virheraportin tunnusmerkkejä. Jotta virheraportti olisi ohjelmistokehittäjälle hyödyllinen, virheen tulee olla toistettavissa ja tarkoin määritelty virheraportissa. Jos kehittäjä ei pysty toistamaan virhettä itse, hän luultavasti hylkää korjausprosessin. Goldberg listaa myös virhetiedot, jotka ohjelmistokehittäjät odottavat saavansa virheraporteissa (Taulukko 1).

GNOME Bug Buddy [16] on GNOME-työpöytäympäristön virheraportointityökalu. Bug Buddy sisältää sekä automaattisen että manuaalisen virheraportoinnin piirteitä: työkalun voi käynnistää joko ohjelmiston kaatuessa tai manuaalisesti. Kaatuneesta ohjelmasta Bug Buddy kerää suorituspinon ja muistivedoksen automaattisesti. Lisäksi Bug Buddy pyytää raportoijaa täyttämään virhettä kuvailevan lomakkeen, johon täytetään manuaalisesti raportavat virhetiedot (Taulukko 1).

Virheraportointi voi tapahtua täysin automaattisesti, jos raportoijan kohtaama virhe aiheuttaa raportointiohjelmistolla havaittavan keskeytyksen. Keskeytys voi johtua esimerkiksi sovellukseen ohjelmoiduista poikkeuksista (engl. *exception*), puutteellisista resursseista käyttöympäristössä tai käyttäjän ohjelma- tai laitteistoympäristön aiheuttamista virheistä [1]. Keskeytyksen havaittuaan raportointiohjelmisto voi kerätä raporttiin erilaisia virhetietoja ja lähettää ne vianseurantajärjestelmään. Kaikki virheet eivät kuitenkaan ole ohjelmallisesti havaittavissa, joten raportoijan on myös pystyttävä raportoimaan virheitä manuaalisesti. Tällöin raportointityötä voidaan helpottaa keräämällä virhetietoa automaattisesti.

Alnefelt ja Malmgren [1] tutkivat yritysten välistä automaattista virheraportointia. He eivät käsitelleet pelkästään aiheen teknistä puolta, vaan heidän tutkimuksensa aihepiiri keskittyi kaikkiaan asiakkaan asenteisiin automaatiota kohtaan, lähetettäviin virhetietoihin, yksityisyyteen, käyttäjän osallistumiseen ja palautteeseen raportoinnista. Tutkimus tehtiin haastattelemalla ohjelmistoyritys IFS:n työntekijöitä ja asiakasyrityksiä. Asenteissa positiivisina asioina nähtiin seuraavat asiat:

– Automaattisella virheraportoinnilla raporttien laatu on parempi, koska raportin täyttäminen ei riipu virheen löytäneen henkilön esitiedoista.

– Automaattinen virheraportointi vähentää työntekijöiden työmäärää, koska raportteja voi lähettää automaattisesti kuka vain.

– Automatisoinnin koettiin nopeuttavan virheenkorjausprosessia.

Asiakasyritykset olivat valmiita lähettämään tietoa IFS:lle vaihtelevasti, joten asiakkaiden kanssa olisi tehtävä kompromisseja virhetiedon määrästä ja tiedonkeruujärjestelmän pitäisi olla lähetettävien virhetietojen suhteen konfiguroitavissa.

Murphy [27] kertoo näkemyksiään automaattisen virheraportoinnin perusperiaatteista järjestelmän onnistumisen kannalta. Raportointilomakkeen on oltava yksinkertainen, häiritsemätön ja yksityisyyttä kunnioittava. Useimpien virheiden korjaukseen riittää pieni määrä automaattisesti kerättäviä virhetietoja (Taulukko 1). Käyttäjän yksityisyys on

otettava huomioon virheitä kerättäessä, vaikka virheiden yksilöiminen eri tahoihin helpottaisi raporttien lajittelua. Murphy huomauttaa myös, että pieni osa ohjelmistojen virheistä aiheuttaa suurimman osan ongelmista. Tästä syystä *Windows Error Reportingissa* [38] virhetiedot analysoidaan ja niille määrätään prioriteetti automaattisesti. Automaattisen virheraportointijärjestelmän täytyy kehittyä ajankuluessa, koska virheprofiilit muuttuvat laitteiston ja ohjelmiston myötä jatkuvasti.

2.3 Yhteenveto

Aikaisemman tutkimuksen perusteella tarve manuaaliselle virheraportoinnille on olemassa, ja nykyisiä vianseurantajärjestelmiä voidaan kehittää virheraportoinnin osalta. Parannukseksi on ehdotettu virheraportoinnin tiedonkeruun automatisointia tai helpottamista. Virhetietoja on pystytty keräämään automaattisesti mm. Windows Error Reportingin [38] avulla, mutta teknologia ei sovellu suoraan manuaaliseen virheraportointiin. Virhetietojen hyödyllisyyttä ohjelmistovirheiden korjaukseen ei ole tutkittu teollisuudessa.

3 Tutkimusaineisto ja -menetelmät

Työssä tehtiin uutta tietoa kartoittava kyselytutkimus viiden eri ohjelmistoyrityksen ohjelmistokehittäjille. Jotta tutkimusaineisto olisi mahdollisimman monipuolinen, kyselyssä käytettiin sekä suljettuja tiettyihin vastausvaihtoehtoihin rajoitettuja että avoimia rajaamattomia kysymyksiä.

Kyselytutkimusmenetelmä soveltui hyvin työn tutkimuskysymyksiin nähden:

1. Työn ensimmäisellä tutkmuskysymyksellä pyrittiin kartoittamaan kehittäjille hyödyllisiä virhetietoja. Useita väitteitä hyödyllisistä virhetiedoista on tehty yksittäisten kehittäjien osalta [15, 16], mutta vain Bettenburg ym. [8] ovat tutkineet tietotarpeita tieteellisin menetelmin. Kyselytutkimusmenetelmällä varmistetaan tulosten yleistettävyys, koska tulosten takana on suuri määrä erilaisia ohjelmistokehittäjiä.

2. Toisella tutkimuskysymyksellä tutkittiin virheraporttien laatua ohjelmistoyritysten ohjelmistotuotannossa. Laadulla tarkoitetaan tässä työssä virheraporttien täydellisyyttä ja virheettömyyttä. Aikaisemmin laatututkimuksia on tehty automaattisilla luokittelutyökaluilla avoimen lähdekoodin ohjelmistokehityksen yhteydessä [8, 30]. Laadun eksakti mittaaminen onnistuisi virhetietokantoja tarkastelemalla, mutta kyselytutkimuksen katsottiin olevan tämän työn tavoitteiden kannalta riittävä.

3. Kolmannella tutkimuskysymyksellä etsittiin tapoja kerätä virhetietoja virheraportteihin. Kyselytutkimuksen lisäksi tarkasteltiin käytössä olevien ohjelmistojen ja työkalujen soveltuvuutta virheraportoinnin tiedonkeruuseen.

3.1 Kyselyn vastaajien valinta

Kyselyn kohderyhmänä olivat ohjelmistokehittäjät viidestä eri yrityksestä (Taulukko 2). Yritykset valittiin tutkimusyhteistyön perusteella ja heidän kehittäjiin otettiin yhteyttä yhteyshenkilöiden kautta saaduilla sähköpostiosoitteilla, joita oli 147. Näin varmistettiin, että kyselyyn vastaajat olivat halutun kohderyhmän sisällä. Kehittäjille pystyttiin lähettämään muistutusviestejä kyselyyn vastaamisesta, minkä on todettu yhdessä tutkimusyhteistyön kanssa nostavan Internet-kyselyiden vastausprosenttia merkittävästi [3]. Kutsusähköpostit lähetettiin kehittäjille LimeSurvey-ohjelmalla [22], jolla kysely toteutettiin.

Taulukko 2: Tiivistelmä kyselyn kohdeyrityksistä. Yrityksestä B erotettiin kaksi kehitysryhmää toimialojen erojen takia.

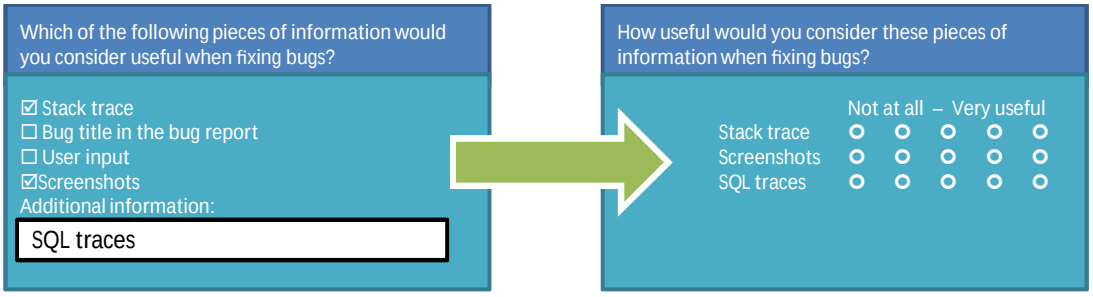
Yritys	Henkilöstö	Liikevaihto (M€/v)	Ikä (v)	Ohjelmistotoimiala
A	50	10	30	Automaatiojärjestelmät
B1	500	50	40	Tekninen suunnittelu
B2	500	50	40	Tietojärjestelmät ja suunnittelu
C	100	10	10	Web-sovellukset
D	100	10	20	Tietojärjestelmät ja simulointi
E	100	10	20	Tekninen suunnittelu ja käyttö

3.2 Kyselyn rakenne

Työn kysely muodostettiin perehtymällä ensin kyselyistä kertovaan kirjallisuuteen [3, 12, 14, 28], minkä jälkeen kysymykset muotoiltiin työn tutkimuskysymyksiä vastaaviksi. Kysymykset ja esivalitut virhetiedot (Liite A) tarkistettiin useaan kertaan tutkimusprojektin sisäisesti, jotta kyselyn kieliasu olisi mahdollisimman selkeä ja yksikäsitteinen.

Kysely koostui kuudesta osuudesta. Ensimmäisessä osuudessa kysyttiin kuinka pitkään vastaaja on työskennellyt nykyisessä työpaikassaan ja yleisesti kehittäjänä, onko hänellä työtehtävissään alaisia ja kuinka monta ohjelmointivirhettä hän on korjannut nykyisessä työpaikassaan. Seuraavissa osuuksissa esitettiin tutkimuskysymyksiin perustuvia kysymyksiä. Kysymykset 2.–5. osuuksissa olivat:

1. *Mitkä tiedot ovat ylipäätään hyödyllisiä virheiden korjaukseen?* Kysymykseen vastattiin valitsemalla virhetietoja esivalitulta listalta (Liite A). Listalla oli yhteensä 18 virhetietoa, jotka valittiin Goldbergin virheraportointiohjeiden [15] pohjalta. Myös Bettenburgin ym. kyselyn [8] virhetiedot perustuivat Goldbergin ohjeisiin. Jos vastaaja ei valinnut virhetietoa tässä kysymyksessä, virhetieto ei toistunut enää seuraavissa kysymyksissä (Kuva 1). Näin varmistettiin, että vastaaja osaa vastata hänelle esitettyihin kysymyksiin. Esimerkiksi jos vastaaja ei pidä laitealustaa tärkeänä tietona virheraporteissa, hän ei todennäköisesti ole seurannut virheitä korjatessaan, ovatko tiedot laitealustasta puuttuneet tai olleet virheellisiä. Kysymykseen pystyi vastaamaan myös kolmella avoimella tekstikentällä, jotta kyselyn vastaukset eivät perustuisi pelkästään esivalittuun listaan.



Kuva 1: Demonstraatio kyselyn toiminnallisuudesta ensimmäisen kysymyksen jälkeen. Valitsemattomat virhetiedot eivät toistuneet seuraavissa kysymyksissä.

2. *Kuinka hyödyllisiä valitsemasi virhetiedot ovat virheiden korjaukseen?* Edellisessä osuudessa valitut virhetiedot arvosteltiin viiden pisteen Likert-asteikolla [12, s. 158–159] ”ei ollenkaan hyödyllistä”–”hyvin hyödyllistä”.

3. *Kuinka usein valitsemissasi virhetiedoissa on esiintynyt puutteita tai virheitä virheraporteissa?* Vastaukset luokiteltiin edellisen kysymyksen tapaan viiden pisteen Likert-asteikolla ”ei koskaan tai hyvin harvoin”–”hyvin usein tai aina”.

4. *Kuinka helppoa valitsemiesi virhetietojen automaattinen kerääminen olisi?* Vastaukset luokiteltiin viiden pisteen Likert-asteikolla ”hyvin vaikeaa”–”hyvin help-

poa”. Vastaaja pystyi myös vastaamaan neutraalisti valinnalla ”en tiedä”. Jotta kysymyksen vastaukset olisivat paremmin hyödynnettävissä, kiinteän asteikon lisäksi automaatiomahdollisuuksia pystyi kommentoimaan avoimella tekstikentällä.

Tässä työssä viitattaessa 1.–4. kysymykseen tarkoitetaan yllä olevia kysymyksiä. Kyselyn viimeisessä kuudennessa osuudessa kerrottiin vastaajalle aikaisemmista tutkimustuloksista ja pyydettiin jättämään aiheeseen liittyviä kommentteja.

Vaikka työn kyselyssä ei lähtökohtaisesti kysytä vastaajien mielipiteitä, vastaukset heijastavat vastaajien subjektiivisia asenteita ja mielipiteitä virheiden korjaukseen liittyen. Coolicanin mukaan [12, s. 156–157, 169–175] hyvässä kyselyssä seuraavat väittämät käyvät toteen:

Erottelevuus: *kyselyn asteikkojen on katettava kaikki mahdolliset vastaukset ja ne täyttävät vastausasteikon tasaisesti.* Väittämä on otettu huomioon työn kyselyssä käytettyjen asteikoiden suunnittelussa.

Luotettavuus: *kyselyn vastaukset ovat sisäisesti johdonmukaisia ja tulokset eivät vaihtele satunnaisesti mitattaessa samaa asiaa.* Luotettavuuden varmistamiseksi kysymysten ja vastausvaihtoehtojen on oltava mahdollisimman yksiselitteisiä ja helposti ymmärrettäviä. On myös varmistettava, että vastaajat pystyvät vastaamaan kysymyksiin tietojensa perusteella. Kyselyssä luotettavuutta pyrittiin lisäämään tarkistuttamalla kysely tutkimusprojektin sisäisesti. Kyselyn ensimmäisessä osuudessa kysyttiin kehittäjän kokemukseen liittyviä kysymyksiä, joilla varmistettiin vastaajan asiaan perehtyneisyys.

Pätevyys: *kysely mittaa haluttua asiaa.* Tätäkin väittämää tuki kyselyn tarkistuttaminen tutkimusprojektin sisällä.

Yleistettävyys: *kyselyn tulokset voidaan yleistää otosta suurempaan ilmiöön.* Kyselyn kohderyhmän koko oli riittävä tulosten yleistettävyyden kannalta. Toisaalta yritysten toimialojen skaala olisi voinut olla suurempi.

3.3 Vastausten analysointi

Tutkimuskysymyksiin vastattiin tarkastelemalla vastausten jakaumia. Likert-asteikollisia muuttujia käsitellään oletusarvoisesti järjestysasteikollisesti, jolloin muuttujan keskittyneisyyden mittana käytetään mediaania ja hajonnan mittana kvartiilivälin pituutta [12, s. 240]. Kvartiilit jakavat vastaukset neljään yhtä suureen osaan. Kvartiiliväli kertoo, mille välille puolet vastaajista ovat vastanneet mediaanin ympärillä. Jotta yritysten väliset erot tulisivat paremmin esiin tulosten tarkastelussa, tarkasteltiin yrityksiä myös erillään.

Virhetietojen vertailemiseksi verrattiin ensin vastausten mediaaneja eri virhetietojen välillä. Jos mediaanit olivat samat, verrattiin 2. ja 4. kvartiilin summia. Jos nekin olivat samat, virhetiedot todettiin samanarvoisiksi kysymyksen mitta-asteikolla. Kvartiilien summaus ei ole järjestysasteikollisille muuttujille tilastollisesti sallittu operaatio, mutta operaation avulla otetaan huomioon vastausten hajonta. Operaatio suosi vastauksia, jotka olivat hajaantuneet vähän tai joiden vastausjakauma oli vinoutunut korkeammalle kuin toisten vastausten.

Koska ensimmäinen ja toinen kysymys vastasivat toisiaan asteikoltaan ja muuttujaltaan, voitiin nämä vastaukset yhdistää hyödyllisten virhetietojen analysointia varten. Vastaajat, jotka eivät valinneet ensimmäisessä kysymyksessä virhetietoa, eivät pitäneet sitä ollenkaan hyödyllisenä, mikä vastasi toisen kysymyksen alhaisinta vastausvaihtoehtoa. Neljännen kysymyksen vastauksia analysoidessa poistettiin neutraalit vastaukset.

Avointen kysymysten vastauksista etsittiin yhteisiä piirteitä. Ensimmäiseen kysymykseen vastatut virhetiedot jaoteltiin luokkiin, jotka muodostettiin vastausten tarkastelun aikana. Neljännen kysymyksen ja kommenttiosion kommentit luokiteltiin kommenteissa havaittavien teemojen mukaan.

4 Tulokset

Kyselykutsun sai yhteensä 142 ohjelmistokehittäjää 5 sähköpostiosoitteen osoittauduttua virheelliseksi tai vanhenneeksi. Kutsutuista 88 avasi ja 74 suoritti kyselyn kokonaan (Taulukko 3). Vastausprosentti oli 52%, jota voidaan pitää erinomaisena lukuna Internet-kyselyssä [28]. Vastausprosenttia olennaisesti kasvatti kutsuviestin jälkeen lähetetyt muistutusviestit.

Taulukko 3: Kutsutut ja vastanneet kehittäjät eri yrityksissä.

Yritys	Kutsutut	Vastanneet	Valmiit	Lisätiedot	Kommentit
A	17	8 (47.1%)	6 (35.3%)	1	2
B1	31	18 (58.1%)	12 (38.7%)	5	4
B2	18	10 (55.6%)	9 (50.0%)	0	3
C	23	20 (87.0%)	18 (78.3%)	8	6
D	31	21 (67.7%)	19 (61.3%)	6	8
E	22	11 (50.0%)	10 (45.5%)	2	2
Yhteensä	142	88 (62.7%)	74 (52.1%)	22	30

Kyselyn vastaajat olivat kokeneita ohjelmistokehittäjiä. Puolet vastaajista oli työskennellyt yrityksessään yli kolme vuotta ja ylipäätään kehittäjänä yli kuusi vuotta. Jokaisella kehittäjällä oli vähintään vuoden työkokemus ohjelmistokehittämisestä. 73% vastaajista oli korjannut nykyisessä työpaikassaan sata tai enemmän ohjelmointivirhettä. Alle 20 ohjelmointivirhettä nykyisessä työpaikassaan oli korjannut 7 henkilöä, joista vain yksi oli työskennellyt yrityksessä alle vuoden. Vastanneista 16 työskenteli esimiesasemassa ja 58 alaisena.

4.1 Kvantitatiiviset tulokset

Kyselyn vastauksista laskettiin mediaanit ja kvartiilit (Liite B), minkä jälkeen virhetiedot järjestettiin suuruusjärjestykseen eri mitta-asteikoilla (Taulukko 4). Vastausten ääripäät olivat tiivistetysti seuraavat:

1. Kehittäjille hyödyllisimmät virhetiedot virheiden korjaukseen olivat *virheen toisto-ohjeet* ja *sovelluksen osa, jossa virhe tapahtui*. Vähiten hyödyllisiksi nähtiin *testitapaukset ja -skriptit* ja *laitealusta*.

2. Eniten ongelmia virheraporteissa oli havaittu *laitealustan, sovelluksen asetusten, käyttäjän vuorovaikutuksen, odotetun käyttäytymisen, virheen toisto-ohjeiden, suorituspinon, ohjelmistoympäristön* ja *kuvakaappausten* kohdalla. Vähiten ongelmia oli *raportoijan yhteystietojen* ja *virheraportin otsikon* kohdalla.

3. *Tuotetiedot, ohjelmistoympäristö* ja *laitealusta* nähtiin helpoimmin kerättäviksi virhetiedoiksi. *Testitapaukset ja -skriptit* ja *odotettu käyttäytyminen* sen sijaan olivat

vaikeasti kerättävissä automaattisesti vastanneiden mukaan. Kaikki virhetiedot voidaan jakaa vastausten perustella joko helposti kerättäviksi tai vaikeasti kerättäviksi automaattisesti.

Vastauksissa esiintyi selvää hajontaa yritysten välillä joidenkin virhetietojen kohdalla. Esimerkiksi *virheen toisto-ohjeet* määriteltiin hyvin hyödyllisiksi jokaisessa yrityksessä, kun taas *suorituspinon* ja *käyttäjän vuorovaikutuksen* hyödyllisyys riippui vastaajan yrityksestä (Liite C). Hajonta oli suurempaa virhetietojen hyödyllisyydessä kuin virheraporttien puutteellisuudessa tai virheellisyydessä tai virhetietojen automaation helppoudessa (Liite D).

Jos vastausten mediaanit summataan yhteen eri mitta-asteikoiden välillä, esille nousevat tutkimuksen kannalta kriittisimmät virhetiedot (summaus ei ole tilastollinen vaan havainnollistava operaatio). Nämä kriittiset virhetiedot ovat hyödyllisiä virheiden korjauksessa, niiden raportoinnissa on ongelmia tai niiden keräämisen automatisointi on helppoa. Kriittisimpiä virhetietoja olivat *sovelluksen osa, jossa virhe tapahtui, sovelluksen asetukset* ja *suorituspino.* Vähiten kriittisiä olivat *virheen vakavuus, virheraportin otsikko* ja *testitapaukset ja -skriptit* (Taulukko 5). Virhetietojen kriittisyys vaihtelee yritysten välillä. Esimerkiksi *virheraporttien* mediaanien summa vaihtelee välillä 8–11 (Liite C).

Taulukko 4: Virhetiedot järjestettyinä eri mitta-asteikoiden mukaan.

1. ja 2. hyödyllisyys

Virhetieto	Mediaani
Virheen toisto-ohjeet	5
Sovelluksen osa	5
Käytetty data	4
Kuvakaappaukset	4
Sovelluksen asetukset	4
Havaittu käyttäytyminen	4
Sovelluksen komponentti	4
Odotettu käyttäytyminen	4
Suorituspino	4
Käyttäjän vuorovaikutus	4
Tuotetiedot	3
Virheraportit	3
Virheraportin otsikko	2
Ohjelmistoympäristö	2
Raportoijan yhteystiedot	2
Virheen vakavuus	2
Testitapaukset ja -skriptit	1
Laitealusta	1

3. puutteellisuus tai virheellisyys

Virhetieto	Mediaani
Laitealusta	4
Sovelluksen asetukset	4
Käyttäjän vuorovaikutus	4
Odotettu käyttäytyminen	4
Virheen toisto-ohjeet	4
Suorituspino	4
Ohjelmistoympäristö	4
Kuvakaappaukset	3.5
Käytetty data	3
Testitapaukset ja -skriptit	3
Sovelluksen komponentti	3
Virheraportit	3
Tuotetiedot	3
Sovelluksen osa	3
Havaittu käyttäytyminen	3
Virheen vakavuus	3
Raportoijan yhteystiedot	2
Virheraportin otsikko	2

4. automaattisen keräämisen helppous

Virhetieto	Mediaani
Tuotetiedot	5
Ohjelmistoympäristö	4
Laitealusta	4
Raportoijan yhteystiedot	4
Suorituspino	4
Sovelluksen osa	4
Sovelluksen komponentti	4
Sovelluksen asetukset	4
Käytetty data	4
Kuvakaappaukset	4
Virheraportit	4
Käyttäjän vuorovaikutus	2
Havaittu käyttäytyminen	2
Virheen toisto-ohjeet	2
Virheraportin otsikko	2
Virheen vakavuus	2
Testitapaukset ja -skriptit	2
Odotettu käyttäytyminen	1

4.2 Kvalitatiiviset tulokset

Ensimmäisen kysymyksen avoimilla tekstikentillä saatiin 22 vastausta (Liite E). Vastaukset tarkasteltiin läpi ja jaettiin yhdeksään eri ryhmään (Taulukko 6). Vastatut virhetiedot ovat esivalitusta listasta poiketen yksikäsitteisempiä ja enemmän sidoksissa yritysten toimialaan.

Taulukko 5: Tärkeimmät virhetiedot järjestettyinä mediaanien summan mukaan.

Virhetieto	Pisteet
Sovelluksen osa	12
Sovelluksen asetukset	12
Suorituspino	12
Kuvakaappaukset	11.5
Tuotetiedot	11
Sovelluksen komponentti	11
Käytetty data	11
Virheen toisto-ohjeet	11
Ohjelmistoympäristö	10
Käyttäjän vuorovaikutus	10
Virheraportit	10
Laitealusta	9
Havaittu käyttäytyminen	9
Odotettu käyttäytyminen	9
Raportoijan yhteystiedot	8
Virheen vakavuus	7
Virheraportin otsikko	6
Testitapaukset ja -skriptit	6

Taulukko 6: Avoimiin kysymyksiin vastatut hyödylliset virhetiedot luokiteltuina.

Virhetieto	Vastaukset (Liite E)	Yritykset
Internet-sivuston osoite	6, 11, 20	C
Käyttäjäprofiili	8, 10, 13, 16, 18	C, D
Tietokantalokit	3, 7, 9, 19	D
Tietoliikennelokit	12	A
Verkon konfiguraatio	5	E
Virheen toisto-ohjeet videona	2, 4	B1
Virheet muissa ohjelmissa	17	E
Virhehistoria	15, 21	B1
Luokittelemattomat virhetiedot	1, 14, 22	–

Avoimilla kysymyksillä saatiin yhteensä 25 kommenttia, joista 19 liittyi virhetietojen automaattiseen keräämiseen ja 11 oli yleisiä tutkimukseen liittyviä kommentteja. Seuraavia tiedonkeruumenetelmiä ehdotettiin (esimerkkikommentit kirjoitettu alkuperäisessä muodossaan):

Helposti poimittavat virhetiedot. *5 kommenttia yrityksiltä B2, C ja D.*

”Stack trace and product information can be read from web server or framework information when there’s uncaught exception.” (Yritys C)

Ohjelmiston käytön nauhoitus. *4 kommenttia yrityksiltä B1, C, D ja E.*

”Just a simple button on each screen to start collecting events and context data or to mark and comment what is happening or what should be happening or what should be changed – –.” (Yritys C)

Raportoijan antamat lisätiedot. *4 kommenttia yrityksiltä A, C ja D.*
”– – This data can be joined to error report page where user can write additional information.” (Yritys C)

Parempien keskeytysten ohjelmointi. *1 kommentti yritykseltä D.*
”Exceptions etc should point to their place of origin and that information should be given in the details of the error message that is shown to the user.” (Yritys D)

Työkalun teknistä toteutusta pohdittiin:

Työkalun integrointi testattavaan järjestelmään. *3 kommenttia yrityksiltä C, D.*
”By a bug reporting form inside the application.” (Yritys C)

Työkalu kolmantena osapuolena. *4 kommenttia yrityksiltä A, C, D ja E.*
”– – the buggy application could supply an other part through a standardized interface.” (Yritys A)

Virheraportoinnissa virhe voidaan havaita joko automaattisesti tai käyttäjä voi aloittaa raportoinnin havaittuaan virheen:

Automaattinen virheraportointi. *8 kommenttia yrityksiltä A, C, D ja E.*
”The used software should detect errors in its execution and then generate an automatic error report with as much information as possible.” (Yritys C)

Käyttäjän aloittama virheraportointi. *4 kommenttia yrityksiltä C ja D.*
”Product itself should create a dump or similar file when user selects ”create bug report” function.” (Yritys D)

Muita kommentteja annettiin tutkimuskysymyksiin liittyen:

Tukipalvelut parantavat virheraporttien laatua. *1 kommentti yritykseltä B1.*
”– – our (developer) bug database contains almost always enough information to proceed fixing the bug. Perhaps this is because there is a front line bug-db in customer services.” (Yritys B1)

Testaajien virheraporttien laatua voidaan parantaa. *1 kommentti yritykseltä B1.*
”Not even in-house testers are able to produce good enough bug reports.” (Yritys B1)

Virheet ovat kytköksissä viitekehykseensä. *2 kommenttia yrityksiltä B2 ja C.*
”Context is essential – what user is trying to do with the system. Sometimes bugs are more complex and tightly associated with the data context.” (Yritys C)
”Different kind of bugs require different input for fixing.” (Yritys B2)

Virheraportointia pystytään parantamaan. *2 kommenttia yrityksiltä C ja D.*
”Automatic information collection can probably help some, in some cases it would be extremely useful.” (Yritys D)

Kaikkea tietoa ei voida kerätä automaattisesti. *2 kommenttia yrityksiltä B2 ja C.*

”Automatically collected information about system and software environment sounds good, but still user should answer at least 1) what she/he was trying to do and 2) how and 3) what happened.” (Yritys B2)

”In web based application user input and screenshots are hard to get.” (Yritys C)

Asiakkaalta ei voida odottaa suurta työpanosta virheraportointiin. *1 kommentti yritykseltä D.*

”What can you expect from the end customer depends, automatic data collection might be most useful in this respect. In general, it is probably unrealistic to assume that the end customer is willing to put a lot of effort in collecting the data, or even to know what data to collect.” (Yritys D)

5 Tarkastelu

Tässä luvussa vastataan johdannossa esitettyihin työn tutkimuskysymyksiin saatujen tulosten pohjalta sekä pohditaan tulosten luotettavuutta, pätevyyttä ja yleistettävyyttä.

5.1 Mitkä virhetiedot ovat ohjelmistokehittäjille hyödyllisiä ohjelmointivirheiden korjauksessa?

Virheiden korjaukseen hyödyllisimmiksi koetut virhetiedot olivat *virheen toisto-ohjeet* ja *sovelluksen osa, jossa virhe tapahtui*. Virheen toisto-ohjeet on myös mainittu ylimääräisissä annetuissa tiedoissa ja automaattisen tiedonkeruun toteutukseen liittyvissä kommenteissa. Myös Bettenburgin ym. [8] tutkimuksessa virheen toisto-ohjeet oli tärkein kehittäjien tarvitsema virhetieto.

Tulos ei ole kovin yllättävä, koska kyseiset virhetiedot ovat virheen määrittelyn kannalta olennaisimpia mm. ohjelmistokehittäjä Tathamin mukaan [36]. Hän kertoo, että eräs parhaista tavoista raportoida virhe on näyttää kehittäjälle, mikä ohjelman toiminnassa on virheellistä ja miten virheellinen toiminta saadaan aikaiseksi. Käsitteet ovat myös hyvin monimerkityksellisiä. Esimerkiksi virheen toisto-ohjeet voivat sisältää muita esivalitun listan virhetietoja, kuten käytetyn datan.

Joidenkin virhetietojen hyödyllisyydessä esiintyi selvää hajontaa yritysten välillä. Yritysten toimialojen erot näkyivät esivalittuun listaan perustuvien vastausten mediaaneissa (Liite C), ja ylimääräiset annetut virhetiedot liittyivät yrityksen toimialaan tai ohjelmiston suoritusympäristöön (Taulukko 6). Myös vastausten kommenttien perusteella virheitä on erilaisia sovelluksen kaatumisesta käytettävyysongelmiin, ja virheen viitekehys on olennainen asia hyödyllisten virhetietojen määrittelemisessä. Joissakin vastauksissa oli hajontaa myös yritysten sisällä (Liite B). Tämä voi johtua siitä, että yritysten sisälläkin kehittäjien tehtävät ja siten korjattavat virheet eroavat toisistaan. Sekä kvantitatiivisten että kvalitatiivisten tulosten perusteella voidaan perustellusti väittää, että virheiden korjauksessa hyödylliset virhetiedot riippuvat virheestä ja ohjelmiston suoritusympäristöstä.

5.2 Missä ohjelmistokehittäjille hyödyllisissä virhetiedoissa on ollut puutteita tai virheitä virheraporteissa?

Ongelmallisimmat virhetiedot virheraporteissa olivat *laitealusta, sovelluksen asetukset, käyttäjän vuorovaikutus, odotettu käyttäytyminen, virheen toisto-ohjeet, suorituspino, ohjelmistoympäristö* ja *kuvakaappaukset*. Lähes kaikki virhetiedot olivat olleet virheraporteissa joskus puutteellisia tai virheellisiä. *Raportoijan yhteystiedoissa, virheraportin otsikossa* ja *virheen vakavuudessa* kerrottiin olevan vähiten ongelmia. Nämä tiedot syötetään usein vianseurantajärjestelmien virheraportoinnissa omiin kenttiinsä [10, 18], ja ne voidaan nähdä melko helposti raportoitavina virhetietoina. Suurena ongelmana voidaan nähdä tärkeiksi koetut, mutta melko usein ongelmalliset *virheen toisto-ohjeet*. Myös *kuvakaappaukset, sovelluksen asetukset, odotettu käyttäytyminen, suorituspino* ja *käyttäjän vuorovaikutus* olivat määritelty keskimääräistä hyödyllisemmiksi virheiden korjaukseen ja niissä esiintyi keskimääräistä enemmän puuttellisuuksia tai virheellisyyksiä virheraporteissa. Tulosten perusteella voidaan väittää, että ohjelmistokehittäjät eivät saa kaikkia

hyödyllisiä virhetietoja virheraporteissa. Vastaukset eivät eronneet merkittävästi yritysten välillä, joten ongelmia esiintyy toimialasta riippumatta.

Kommenttiosiosta saaduissa kommenteissa oli virhetietojen puutteellisuuteen tai virheellisyyteen liittyen kaksi kommenttia. Toisessa kommentissa kerrottiin, että yrityksen virheraportit ovat melkein aina tarpeeksi hyviä virheiden korjaukseen, minkä selittäisi se, että yrityksen sisäinen tukiorganisaatio muodostaa asiakkailta tulleet raportit kehittäjille. Toisaalta toinen saman yrityksen kehittäjä kommentoi, että yrityksen sisäiset testaajat eivät saa aikaiseksi tarpeeksi hyviä virheraportteja. Tämä osoittaisi, että tukihenkilöstön kouluttaminen virheraportointiin ei välttämättä poistaisi ongelmaa kokonaan, koska raportteja tulee monesta eri lähteestä (asiakkaat, testaajat, eri työntekijäryhmät [24]) virhetietokantoihin, joista kehittäjät saavat tiedon korjattavista virheistä.

5.3 Miten kehittäjille hyödyllisiä virhetietoja voidaan kerätä virheraportteihin mahdollisimman vaivattomasti?

Kvantitatiivisten vastausten perusteella virhetiedot jakautuvat seuraavasti kahteen ryhmään (Taulukko 4):

Helppo kerätä automaattisesti:

- Tuotetiedot
- Ohjelmistoympäristö
- Laitealusta
- Raportoijan yhteystiedot
- Suorituspino
- Sovelluksen osa
- Sovelluksen komponentti
- Sovelluksen asetukset
- Käytetty data
- Kuvakaappaukset
- Virheraportit

Vaikea kerätä automaattisesti:

- Käyttäjän vuorovaikutus
- Havaittu käyttäytyminen
- Virheen toisto-ohjeet
- Virheraportin otsikko
- Virheen vakavuus
- Testitapaukset ja -skriptit
- Odotettu käyttäytyminen

Luokittelu havaitaan myös kvalitatiivisissa vastauksissa. Kommenttien perusteella osa virhetiedoista on helppo kerätä automaattisesti ja niiden suhteen virheraportointia pystytään parantamaan nykyisestään. Kaikkea virhetietoa ei ole mahdollista kerätä automaattisesti, mutta myös vaikeasti kerättävien virhetietojen keräämistä voidaan helpottaa esimerkiksi ohjelmiston käyttöä nauhoittamalla tai pyytämällä raportoijalta lisätietoja.

Luokittelu vastaa automaattisessa virheraportoinnissa käytettyjä virhetietoja (Taulukko 1). Vastauksissa ilmenevien helposti kerättävien virhetietojen lisäksi *käyttäjän vuorovaikutus* ja *testitapaukset* pystytään keräämään automaattisesti aikaisemman tutkimuksen perusteella [1, 5]. Kuten kommenteissa todettiin, *käyttäjän vuorovaikutuksen* kerääminen on vaikeaa esimerkiksi web-sovellusten yhteydessä. Vaikka *testitapausten* automaattinen kerääminen onkin mahdollista [5], menetelmä ei ole välttämättä kovin tunnettu, joten kyselyn tulos ei ole yllättävä. Tulosten voidaan katsoa siten olevan johdonmukaisia suhteessa aiempaan.

Helposti kerättävistä virhetiedoista erityisesti *sovelluksen osa, sovelluksen asetukset, suorituspino, kuvakaappaukset, tuotetiedot, sovelluksen komponentti* ja *käytetty data* tulisi kerätä automaattisesti virheraportteihin. Nämä kriittiset virhetiedot ovat hyödyllisiä virheiden korjauksessa ja ovat olleet puutteellisia tai virheellisiä virheraporteissa (Taulukko 5). Kerättävien virhetietojen määrittelyssä tulee ottaa huomioon myös ohjelmiston suoritusympäristö, koska esimerkiksi yrityksessä E *virheraportit* ovat myös kriittisiä virhetietoja ja ne tulisi myös kerätä automaattisesti (Liite B).

Kommenttien perusteella virhetietojen keräämisen toteutuksessa tulee ottaa huomioon seuraavat asiat:

– Automaattinen virheraportointi on mahdollista, mutta myös manuaalisen virheraportoinnin mahdollisuus pitää säilyttää.

– Integroidaanko keräämistyökalu kohdeohjelmistoon vai suoritetaanko kerääminen kolmannen osapuolen työkalulla?

Asioita pohditaan seuraavaksi tarkemmin.

Automaattisella virheraportoinnilla pystytään keräämään ohjelmistosta helposti kerättävät virhetiedot, mutta automaatio vaatii ohjelmallisesti tunnistettavan poikkeuksen. Kyselyn vastauksissa mainittiin parempien keskeytysten ohjelmoinnin helpottavan virheraportointia. Poikkeuksen tapahtuessa sovelluksesta pystytään keräämään jopa testitapaus automaattisesti [5]. Jos poikkeusta ei tapahdu, automaattinen virheen sijainnin tarkka määrittäminen on mahdotonta [23], minkä takia tarve manuaaliselle virheraportoinnille on olemassa. Tällaisissa tilanteissa joitakin helposti kerättäviä virhetietoja voidaan kerätä virheraporttiin, mutta käyttäjän tiedonkeruuta on myös helpotettava muilla keinoin.

Vaikeasti kerättävät virhetiedot kiteytyvät ohjeisiin virheen toistamiseen. Toisto-ohjeiden avulla ohjelmistokehittäjä pystyy toistamaan virheen itse ja ymmärtää, mikä ohjelmiston toiminnassa on virheellistä. Kyselyn kommentteissa mainittu ohjelmiston käytön nauhoittaminen mahdollistaisi toisto-ohjeiden raportoimisen vaivattomasti kehittäjille.

Ohjelmiston käyttöä voidaan nauhoittaa esimerkiksi tallentamalla tietokoneen näytön sisältö videotiedostoksi tai ohjelmiston tilan muutokset voidaan havaita ja tallentaa [5]. Videotiedoston nauhoittamisen toteuttaminen olisi teknisesti helppoa ja siihen löytyy valmiita *Screencast*-ohjelmistoja [31], mutta videotiedostojen suuri koko voi muodostua ongelmaksi raporttien varastoinnissa ja lähettämisessä. Nykyisissä vianseurantajärjestelmissä on mahdollista lähettää videotiedostoja raportissa. Ohjelmiston tilan muutosten nauhoittaminen on teknisesti haastavampaa ja vaatii työkalun määrittämisen ohjelmiston suoritusympäristön mukaan. Valmiita työkaluja on tähänkin saatavilla [26, 32, 34], mutta kyselyn tulosten perusteella niitä ei käytännössä käytetä. Kyseisten työkalujen käyttö vaatisi raportoijien kouluttamista, eikä työkaluja ole aina integroitu vianseurantajärjestelmiin.

Esimerkiksi Microsoft Visual Studio Test Professional 2010 -ohjelmistolla [26] testaaja pystyy nauhoittamaan uudelleen toistettavia ohjelmasuorituksia. Suoritusten avulla kehittäjälle voidaan raportoida myös uudelleen toistumattomat virheet [37]. Ohjelmisto liittää virheisiin myös muita virhetietoja kuten laitealustan ja käyttäjä voi lisätä raporttiin odotetun käyttäytymisen ja esimerkiksi kuvakaappauksia. Toisaalta ohjelmisto on rajoitettu vain yhteen ohjelmistokehitysympäristöön, eikä se ole ohjelmiston loppukäyttäjän käytettävissä.

Viimeiseksi on syytä mainita kommenteissa esitetty näkökulma teknisestä toteutuksesta ohjelmiston loppukäyttäjän ja testaajan välillä. Loppukäyttäjältä ei voida vaatia suurta työpanosta virheraportointiin, koska hän olettaa ohjelmiston täyttävän kaikki vaatimukset ja toimivan moitteettomasti. Vastuu ohjelmiston laadun parantamisesta on näin katsoen ohjelmiston testaajalla. Koska testaajan voidaan olettaa olevan halukkaampi käyttämään enemmän aikaa raportointiin, hänet tulisi kouluttaa edistyneen raportointityökalun käyttämiseen. Loppukäyttäjälle paremmin soveltuvat automaattinen virheraportointi tai kevyehkö raportointityökalu integroituna ohjelmistoon tai käyttöjärjestelmään.

5.4 Tulosten luotettavuus

Osa vastaajista jätti kyselyn kesken, mikä on tyypillistä Internet-kyselyissä [28]. Mitään selittävää muuttujaa kesken jättämiselle ei löytynyt tarkastelemalla vastauksia. Vastausten analysointiin käytettiin vain kyselyn kokonaan suorittaneita vastauksia aineiston luotettavuuden varmentamiseksi, paitsi ensimmäisen kysymyksen avoimen osuuden vastauksista käytettiin myös keskeneräisiä vastauksia.

Vastaajan motivaation ylläpitämisen takia on tärkeää miettiä, onko vastaajan vastattava jokaiseen kysymykseen jollakin vaihtoehdolla [28]. Tässä kyselyssä vain neljännessä kysymyksessä oli mahdollista vastata neutraalilla tavalla olematta mitään mieltä asiasta. Kolmen ensimmäisen osuuden kysymykset nähtiin yksinkertaisina vastaamisen kannalta, joten neutraalia vaihtoehtoa ei näihin osuuksiin sisällytetty. Näin toimimalla varmistettiin vastaajien osallistuminen, mutta saatettiin vääristää vastaajien todellisia mielipiteitä. Neljännessä kysymyksessä neutraali vaihtoehto sisällytettiin, koska kysymykseen vastaaminen ei ollut niin yksiselitteistä kuin aiempiin kysymyksiin.

Bettenburgin ym. [8] tutkimuksessa kyselyyn kutsuttiin vain kehittäjiä, joille oltiin määrätty vähintään 50 ohjelmointivirhettä korjattaviksi. Tällä toimenpiteellä varmistettiin, että kyselyyn vastasivat vain kokeneet kehittäjät. Tämän työn kyselyssä kysyttiin kehittäjien työkokemus vuosina ja kuinka monta ohjelmointivirhettä he ovat korjanneet nykyisessä työpaikassaan. Ohjelmointivirheiden määräämistä ei voida katsoa yhdenvertaiseksi korjauksen kanssa, koska esimerkiksi Eclipse-ohjelmiston virheraporteista vain 40% oli korjattu Anvikin ym. [4] tutkimuksessa. Vastanneiden kokemuksen perusteella ei nähty tarvetta vastaajamäärän supistamiseen, vaan kaikkien vastaajien nähtiin olevan tarpeeksi kokeneita kyselyn luotettavuuden kannalta.

5.5 Tulosten pätevyys

Selvitetyistä hyödyllisistä virhetiedoista voidaan päätellä, että toiset ovat yleisesti hyödyllisiä ohjelmistokehityksessä ja toiset hyödyllisiä virheestä ja ohjelmiston suoritusympäristöstä riippuen. Toisaalta esimerkiksi laitealustaa ei koettu kyselyn perusteella hyödylliseksi, vaikka se on selvästi olennainen virhetieto esimerkiksi laiteajureiden yhteydessä. Tämän takia tutkimuksen perusteella ei voida sanoa, mitkä virhetiedot olisivat kehittäjille hyödytöntä, koska kohdeyritysten toimialat eivät kattaneet kaikkea mahdollista ohjelmistokehitystä. Kysymyksenasettelusta johtuen virhetiedon hyödyllisyyden mitta voi olla kyseenalainen, koska jotkin virhetiedot voivat olla hyödyllisiä vain tietynlaisten virheiden

korjauksessa. Eksaktimpia vastauksia olisi voinut tutkia esimerkiksi asettelulla: ”Kuinka usein virhetieto olisi hyödyllinen virheiden korjauksessa?”

Virheraporttien puutteellisuuden tai virheellisyyden tutkimisen tarkoituksena oli kartoittaa ongelmallisten virheraporttien määrä. Koska puutteellisista tai virheellisistä virhetiedoista kysyttäessä vastasivat vain ne kehittäjät, jotka olivat kokeneet virhetiedon tärkeäksi, vastaukset puutteiden ja virheiden esiintymisen puolesta implikoivat olemassaolevia ongelmia. Toisaalta vastaukset eivät kerro, oliko virhetieto ollut hyödyllinen silloin kun se puuttui tai oli virheellinen.

Esivalittu lista vähensi vastaajan työtä, mutta myös vaikutti saatuihin vastauksiin. Listalle valitut virhetiedot olivat melko yleisluontoisia asioita, mikä vähensi eroja yritysten välillä. Avoimella tekstikentällä saadut virhetiedot olivat siten paljon yksiselitteisempiä, mutta niissä vastausten otoskoko jäi pieneksi. Olisi mielenkiintoista tehdä kysely uudestaan yksiselitteisemmillä virhetiedoilla.

5.6 Tulosten yleistettävyys

Kyselyn otoskoko oli tarpeeksi suuri johtopäätösten tekemiseen. Vastausprosentti oli korkea, joten vastaajat edustivat valittua vastaajajoukkoa tarpeeksi hyvin. Tulokset eivät välttämättä sovellu kaikilta osin mihin tahansa ohjelmistokehitykseen, vaan hyödylliset virhetiedot riippuvat siitä suuresti. Virheraporttien laatu vaihtelee riippuen yrityksissä käytettävistä menetelmistä. Automaation mahdollisuudet riippuvat ohjelmiston suoritusympäristöstä.

Virheraportoinnin toteutuksessa olisi teknisen näkökulman lisäksi otettava huomioon käyttäjän yksityisyys. Esimerkiksi käytetty data saattaa sisältää arkaluonteista tietoa ohjelmiston käyttäjälle, mikä rajoittaisi kerättävän virhetiedon määrää.

6 Johtopäätökset

Tulosten perusteella vianseurantajärjestelmiä tulisi kehittää virheraporttien laadun parantamiseksi ja siten ongelmien vähentämiseksi ohjelmistotuotannossa. Useiden tärkeiksi koettujen virhetietojen kohdalla oli ongelmia ja vastaajien mukaan ne voitaisiin kerätä melko helposti automaattisesti virheraportteihin.

Hyödyllisimpiä virhetietoja kehittäjille virheiden korjauksessa olivat *virheen toisto-ohjeet* ja *sovelluksen osa, jossa virhe tapahtui*. Joidenkin virhetietojen hyödyllisyydestä kehittäjät olivat yksimielisiä, kun taas joidenkin hyödyllisyyden arvioinnissa esiintyi enemmän hajontaa. Virhetiedon hyödyllisyys riippuukin virheestä ja ohjelmiston suoritusympäristöstä.

Ongelmallisimpia virhetietoja virheraporteissa olivat *laitealusta, sovelluksen asetukset, käyttäjän vuorovaikutus, odotettu käyttäytyminen, virheen toisto-ohjeet* ja *suorituspino*. Virheraporttien virhetiedoissa oli ollut puutteellisuutta tai virheellisyyttä yrityksestä riippumatta. Kehittäjien kommenttien perusteella sisäisen tuen muodostamat raportit parantavat asiakkailta tulevien virheraporttien laatua, mutta yrityksen sisäisten testaajien raporttien laatu on silti vaihtelevaa.

Kehittäjien mielestä esimerkiksi *tuotetiedot, ohjelmistoympäristö* ja *laitealusta* olisi helppo kerätä automaattisesti virheraportteihin, mutta esimerkiksi *havaittu käyttäytyminen* ja *virheen toisto-ohjeiden* täysin automaattinen kerääminen olisi vaikeaa. Erityisesti *sovelluksen osa, sovelluksen asetukset, suorituspino, kuvakaappaukset, tuotetiedot, sovelluksen komponentti* ja *käytetty data* tulisi kerätä automaattisesti virheraportteihin, koska ne ovat hyödyllisiä virheiden korjauksessa ja ovat olleet puutteellisia tai virheellisiä virheraporteissa (Taulukko 5).

Osa virheraportoinnista voidaan automatisoida täysin, mutta myös manuaalista virheraportointia tarvitaan virheisiin, jotka eivät aiheuta tietokoneellisesti havaittavaa poikkeusta. Täysin automaattinen virheraportointi sopii parhaiten ohjelmiston loppukäyttäjille, koska loppukäyttäjiltä ei voida olettaa suurta työpanosta virheraportointiin.

Virheen toisto-ohjeiden keräämistä voidaan helpottaa ohjelmiston käytön nauhoitustyökaluilla. Valmiita työkaluja löytyy koulutettujen testaajien käyttöön [26, 32, 34], mutta niiden käyttämistä varten tulisi kouluttaa yrityksissä myös muita sidosryhmiä. Nauhoitustyökalun tulisi olla helposti käytettävissä yhdessä vianseurantajärjestelmän kanssa, mikä toteutuu tällä hetkellä parhaiten ohjelmiston käyttöä nauhoittamalla *Screencast*-ohjelmistojen [31] avulla.

Viitteet

[1] Alnefelt, P. ja Malmgren, P. Automated error reporting: Business-to-business aspects to consider for a software provider. Master's Thesis, Linköping University, Department of Management and Engineering, Linköping, 2009.

[2] Andersson, C. ja Runeson, P. Verification and validation in industry – a qualitative survey on the state of practice. Teoksessa: *Proceedings of the 2002 International Symposium on Empirical Software Engineering,* 2002, s. 37–47.

[3] Andrews, D., Nonnecke, B. ja Preece, J. Electronic Survey Methodology: A Case Study in Reaching Hard-to-Involve Internet Users. *International Journal of Human-Computer Interaction,* vol. 6, nro 2, 2003, s. 185–210.

[4] Anvik, J., Hiew, L. ja Murphy, G. C. Coping with an open bug repository. Teoksessa: *Proceedings of the 2005 OOPSLA Workshop on Eclipse Technology eXchange,* 2005, s. 35–39.

[5] Artzi, S., Kim, S. ja Ernst, M. ReCrash: Making Software Failures Reproducible by Preserving Object States. Teoksessa: *Proceedings of the 22nd European conference on Object-Oriented Programming,* 2008, s. 542–565.

[6] Bach, J. Test Automation Snake Oil. Verkkodokumentti. Päivitetty 6.3.1999. Viitattu 4.8.2010. Saatavissa: `http://www.satisfice.com/articles/test_automation_snake_oil.pdf`.

[7] Berner, S., Weber, R. ja Keller, R. K. Observations and lessons learned from automated testing. Teoksessa: *Proceedings of the 27th International Conference on Software Engineering,* 2005, s. 571–579.

[8] Bettenburg, N., Just, S., Schröter, A., Weiss, C., Premraj, R. ja Zimmermann, T. What makes a good bug report? Teoksessa: *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering,* 2008, s. 308–318.

[9] Boehm, B. Get Ready for Agile Methods, with Care. *Computer,* vol. 35, nro 1, 2002, s. 64–69.

[10] Bugzilla-ohjelmisto. Viitattu 4.8.2010. Saatavissa: `http://www.bugzilla.org/`.

[11] Burnstein, I. *Practical Software Testing.* New York, Springer, 2003.

[12] Coolican, H. *Research Methods and Statistics in Psychology.* London, Hodder & Stoughton, 1999.

[13] Čubranić, D. Automatic bug triage using text categorization. Teoksessa: *Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering,* 2004, s. 92–97.

[14] Foddy, W. *Constructing Questions for Interviews and Questionnaires: Theory and Practice in Social Research.* Cambridge, Cambridge University Press, 1994.

[15] Goldberg, E. Bug Writing Guidelines. Verkkodokumentti. Viitattu 4.8.2010. Saatavissa: `https://bugs.eclipse.org/bugs/bugwritinghelp.html`.

[16] Gwynne, T., Baudais, E. ja McCance, S. GNOME Bug Buddy – Bug Report Tool. Verkkodokumentti. Viitattu 4.8.2010. Saatavissa: `http://library.gnome.org/devel/bug-buddy/2.14/index.html.en`.

[17] Jacobson, I. ja Bylund, S. *The Road to the Unified Software Development Process.* Cambridge, Cambridge University Press, 2000.

[18] JIRA – Bug, Issue and Project Tracking for Software Development. Viitattu 4.8.2010. Saatavissa: `http://www.atlassian.com/software/jira/`.

[19] Just, S., Premraj, R. ja Zimmermann, T. Towards the next generation of bug tracking systems. Teoksessa: *IEEE Symposium on Visual Languages and Human-Centric Computing,* 2008, s. 82–85.

[20] Kitchenham, B. A., Pfleeger, S. L., Pickard, L. M., Jones, P. W., Hoaglin, D. C., El Emam, K. ja Rosenberg, J. Preliminary guidelines for empirical research in software engineering. *IEEE Trans. Software Eng.,* 2002, vol. 28, nro 8, s. 721–734.

[21] LaToza, T. D., Venolia, G. ja DeLine, R. Maintaining mental models: A study of developer work habits. Teoksessa: *Proceedings of the 28th International Conference on Software Engineering,* 2006, s. 492–501.

[22] LimeSurvey-ohjelmisto. Viitattu 4.8.2010. Saatavissa: `http://www.limesurvey.org/`.

[23] Liu, C., Yan, X., Yu, H., Han, J. ja Yu, P. S. Mining behavior graphs for "backtrace" of noncrashing bugs. Teoksessa *Proceedings of the Fifth SIAM International Conference on Data Mining,* 2005, s. 286–297.

[24] Mäntylä, M. V., Iivonen, J. ja Itkonen, J. Who Tested My Software – An Industrial Case Study of Organization, Values, and Distribution of Testing. Tarkastuksessa, ESEM, 2010.

[25] Marick, B. When should a test be automated. Teoksessa: *Proceedings of the 11th International Software/Internet Quality Week,* 1998.

[26] Microsoft Visual Studio Test Professional 2010. Viitattu 4.8.2010. Saatavissa: `http://www.microsoft.com/visualstudio/fi-fi/products/2010-editions/test-professional`.

[27] Murphy, B. Automating Software Failure Reporting. *Queue,* vol. 2, nro 8, 2004, s. 42–48.

[28] Punter, T., Ciolkowski, M., Freimut, B. ja John, I. Conducting On-line Surveys in Software Engineering. Teoksessa: *Proceedings of the 2003 International Symposium on Empirical Software Engineering,* 2003, s. 80–88.

[29] The R Project for Statistical Computing. Viitattu 4.8.2010. Saatavissa: `http://www.r-project.org/`.

[30] Schugerl, P., Rilling, J. ja Charland, P. Mining Bug Repositories – A Quality Assessment. Teoksessa: *Proceedings of the 2008 International Conference on Computational Intelligence for Modelling Control & Automation,* 2008, s. 1105–1110.

[31] Screencast-ohjelmistoja. Viitattu 4.8.2010. Saatavissa: `http://en.wikipedia.org/wiki/Screencast#Software`.

[32] Selenium web application testing system. Viitattu 4.8.2010. Saatavissa: `http://seleniumhq.org/`.

[33] Sha, L. Using Simplicity to Control Complexity. *IEEE Softw.,* vol. 18, nro 4, 2001, s. 20–28.

[34] Squish – The cross-platform GUI test automation tool. Viitattu 4.8.2010. Saatavissa: `http://www.froglogic.com/products/`.

[35] Subramaniam, B. Effective Software Defect Tracking – Reducing Project Costs and Enhancing Quality. *CrossTalk – Journal of Defense Software Engineering,* vol. 12, nro 4, 1999, s. 3–9.

[36] Tatham, S. How to Report Bugs Effectively. Verkkodokumentti. Päivitetty 8.9.2008. Viitattu 4.8.2010. Saatavissa: `http://www.chiark.greenend.org.uk/~sgtatham/bugs.html`.

[37] Windows Error Reporting – Debugging with IntelliTrace. Viitattu 4.8.2010. Saatavissa: `http://msdn.microsoft.com/en-us/library/dd264915.aspx`.

[38] Windows Error Reporting. Viitattu 4.8.2010. Saatavissa: `http://www.microsoft.com/whdc/winlogo/maintain/wer/errclass.mspx`.

[39] Zimmermann, T., Premraj, R., Sillito, J. ja Breu, S. Improving bug tracking systems. Teoksessa: *Companion to the 31st International Conference on Software Engineering,* 2009, s. 247–250.

A Kyselyn rakenne

Liitteessä esitetään työssä käytetty kysely alkuperäisessä muodossaan englanninkielisenä. Osassa A.1 esitetään kysytyt kysymykset ja osassa A.2 lista kysymyksiin liittyvistä virhetiedoista.

A.1 Kysymykset

Kysymykset on luokiteltu käytettyjen osuuksien mukaan ja niiden vastausvaihtoehdot ovat esitetty suluissa, jos vastaustapaa rajoitettiin. Virhetietokohtaisiin kysymyksiin vastattiin listan A.2 sekä 2. osuudessa syötettyjen virhetietojen perusteella.

1. Background information
   - How many years have you worked as a developer in the company you are currently working at?
   - How many years have you worked as a developer generally?
   - What is your role in your development group? (Management level (I have subordinates), Employee level (no subordinates))
   - How many bugs have you fixed in the company you are currently working at (approximation is enough)? (0–100)
2. Bug information
   - Which of the following pieces of information would you consider useful when fixing bugs? In this context you can relate bugs to any software you have been developing. Check every item you think could be used even slightly in bug fixing.
   - If you consider something else to be useful, that's not listed above, please type it here. (kolme tekstikenttää)
3. Bug information usefulness
   - How useful (when available) would you consider these pieces of information when fixing bugs? Information is considered missing or incorrect if you need to find it out yourself after the initial bug report. (Not useful at all, Slightly useful, Fairly useful, Quite useful, Very useful)
4. Missing or incorrect information in bug reports
   - How often has there been missing or incorrect information in these items in bug reports assigned to you? (Never or very seldom, Quite seldom, Sometimes, Quite often, Very often or always)
5. Collecting information automatically

   - How easy would it be to automatically collect this information when the bug reporter is using software? For example ‘Systeminfo’ command in Windows XP commandline outputs information of the operating system that could be in principle quite easily automatically collected and attached with defect reports. (Very hard, Quite hard, I don’t know, Quite easy, Very easy)
   - If the automation of information collecting is possible, how could it be done?

6. Comments
   - Please feel free to share any interesting thoughts or experiences related to the subject.

A.2 Esivalitut virhetiedot

Kyselyssä käytetyt esivalitut virhetiedot:

- Information from the bug context:
  - Product information
  - Component / module of the application
  - Part of the application where bug occurred, e.g., print dialog
  - Software context, e.g., operating system, user applications
  - Hardware context, e.g., Mac, PC, mobile
  - Configuration of the application, e.g., options set in preferences
  - Operating data, e.g., user file, datamodel, database
- Information from the reporter:
  - Reporter’s contact information
  - Bug title in the bug report
  - User input, e.g., mouse movement, keystrokes
  - Screenshots
  - Observed behavior of the application
  - Expected behavior of the application
  - Steps to reproduce the bug
  - Stack trace
  - Error reports, e.g., Blue Screen, error code 102
  - Test cases, test scripts
  - Severity of the bug

B Kyselyn kvantitatiiviset vastaukset

Liitteessä esitetään kyselyn vastausten mediaanit ja kvartiilivälien pituudet taulukoituina (Taulukot B1, B2 ja B3). Taulukoissa T on esivalitun listan (Liite A) virhetietoa vastaava numero, N otoskoko, M mediaani ja IQR kvartiilivälin pituus.

Taulukko B1: Vastaukset virhetietojen hyödyllisyydestä.

	A			B1			B2			C			D			E			Yhteensä		
T	N	M	I	N	M	I	N	M	I	N	M	I	N	M	I	N	M	I	N	M	I
1	3	2.0	3	10	4.0	2	7	4	2	12	2.0	3	11	3	3	4	1.0	2	47	3	3
2	3	2.5	4	10	4.0	0	7	4	2	14	4.0	2	15	3	3	8	3.0	2	57	4	2
3	5	4.5	1	12	5.0	1	9	5	0	17	5.0	0	18	5	1	7	3.5	4	68	5	1
4	3	2.0	2	7	2.5	2	5	2	3	12	2.5	2	10	2	1	5	1.5	1	42	2	2
5	2	1.0	2	4	1.0	1	4	1	3	7	1.0	1	6	1	1	2	1.0	0	25	1	1
6	5	3.5	1	10	3.5	1	8	4	2	16	3.5	1	17	4	1	5	1.5	2	61	4	1
7	5	3.0	3	12	5.0	0	9	4	1	14	3.5	3	17	4	2	9	5.0	2	66	4	2
8	4	2.0	2	7	1.5	2	5	2	3	10	2.0	2	10	2	2	7	3.5	3	43	2	2
9	3	1.5	1	12	3.5	1	7	3	2	6	1.0	2	11	2	3	8	2.5	1	47	2	3
10	3	1.5	2	9	4.0	2	7	4	2	14	4.0	1	9	1	3	9	4.0	2	51	4	3
11	6	3.0	2	12	5.0	0	8	4	2	17	4.0	1	17	4	2	10	4.0	1	70	4	2
12	6	4.0	1	8	5.0	4	6	5	4	13	4.0	3	16	5	2	8	4.0	2	57	4	3
13	3	2.0	3	8	4.5	4	5	3	3	12	3.0	3	15	4	1	8	4.0	2	51	4	4
14	6	5.0	0	12	5.0	0	9	5	0	17	5.0	0	18	5	0	10	5.0	0	72	5	0
15	3	2.5	4	8	3.0	3	9	5	1	13	4.0	3	11	2	2	8	4.0	2	52	4	4
16	3	2.5	4	8	3.5	3	6	4	3	13	3.0	4	14	3	2	8	3.5	2	52	3	3
17	1	1.0	0	7	3.0	3	3	1	1	6	1.0	2	11	3	3	7	3.5	4	35	1	2
18	3	1.5	3	8	3.0	2	4	1	2	10	2.0	3	7	1	1	8	2.0	1	40	2	2
	6			12			9			18			19			10			74		

Taulukko B2: Vastaukset virhetietojen puutteellisuudesta tai virheellisyydestä.

	A			B1			B2			C			D			E			Yhteensä		
T	N	M	I	N	M	I	N	M	I	N	M	I	N	M	I	N	M	I	N	M	I
1	3	3.0	0	10	3.0	1	7	2.0	0	12	2.0	2	11	3.0	1	4	3.0	0	47	3.0	1
2	3	3.0	0	10	3.5	1	7	2.0	1	14	2.5	1	15	3.0	1	8	3.0	0	57	3.0	2
3	5	3.0	1	12	2.5	1	9	2.0	1	17	3.0	1	18	3.0	2	7	3.0	0	68	3.0	1
4	3	2.0	1	7	4.0	0	5	4.0	1	12	3.5	2	10	4.5	1	5	3.0	2	42	4.0	2
5	2	4.5	1	4	4.0	2	4	3.5	1	7	5.0	2	6	3.5	2	2	3.0	4	25	4.0	3
6	5	2.0	2	10	4.0	0	8	3.0	1	16	4.0	2	17	4.0	1	5	4.0	1	61	4.0	1
7	5	3.0	1	12	3.0	1	9	3.0	1	14	3.5	2	17	3.0	0	9	3.0	2	66	3.0	1
8	4	2.5	1	7	1.0	1	5	1.0	0	10	3.0	2	10	2.0	1	7	1.0	0	43	2.0	2
9	3	3.0	0	12	2.0	2	7	2.0	0	6	2.5	2	11	2.0	0	8	1.5	2	47	2.0	2
10	3	2.0	1	9	4.0	1	7	3.0	0	14	4.0	1	9	3.0	1	9	4.0	0	51	4.0	1
11	6	3.5	2	12	3.5	1	8	2.5	2	17	3.0	1	17	4.0	1	10	4.0	1	70	3.5	1
12	6	3.0	0	8	3.0	0	6	2.0	0	13	2.0	1	16	3.0	1	8	3.0	1	57	3.0	1
13	3	3.0	0	8	4.0	0	5	3.0	1	12	3.0	1	15	4.0	1	8	4.0	0	51	4.0	1
14	6	3.5	1	12	4.0	1	9	3.0	1	17	4.0	1	18	3.5	1	10	3.5	2	72	4.0	1
15	3	2.0	1	8	5.0	0	9	3.0	2	13	2.0	3	11	5.0	2	8	3.5	1	52	4.0	3
16	3	2.0	1	8	3.0	0	6	2.5	1	13	3.0	2	14	3.0	1	8	3.5	1	52	3.0	2
17	1	3.0	0	7	5.0	1	3	3.0	0	6	4.0	1	11	3.0	1	7	3.0	1	35	3.0	1
18	3	3.0	0	8	3.0	1	4	2.5	1	10	2.5	1	7	3.0	1	8	2.5	1	40	3.0	1
	6			12			9			18			19			10			74		

Taulukko B3: Vastaukset automaattisen tiedonkeruun mahdollisuudesta.

	A			B1			B2			C			D			E			Yhteensä		
T	N	M	I	N	M	I	N	M	I	N	M	I	N	M	I	N	M	I	N	M	I
1	2	4.5	1	8	4.5	1	6	4.5	1	12	5.0	1	8	4.5	1	3	4.0	0	39	5	1
2	2	4.5	1	9	4.0	0	6	2.0	2	13	4.0	1	14	4.0	2	6	3.0	3	50	4	2
3	4	4.0	1	9	2.0	2	7	2.0	1	15	4.0	1	12	4.0	2	5	4.0	2	52	4	2
4	3	5.0	0	7	4.0	1	4	5.0	0	9	4.0	0	7	4.0	0	4	4.5	1	34	4	1
5	2	5.0	0	4	4.0	0	3	5.0	0	5	4.0	0	5	4.0	0	2	4.5	1	21	4	1
6	3	4.0	0	9	4.0	0	7	4.0	1	13	2.0	2	15	4.0	2	3	4.0	0	50	4	2
7	3	4.0	0	9	4.0	2	7	4.0	2	10	3.0	2	14	4.0	2	6	4.0	0	49	4	2
8	4	5.0	0	5	4.0	2	2	4.0	0	8	4.5	1	5	5.0	1	7	5.0	1	31	4	1
9	2	5.0	0	9	1.0	1	5	1.0	1	5	4.0	3	9	2.0	3	6	2.5	3	36	2	3
10	2	4.5	1	8	3.0	2	5	2.0	2	11	2.0	0	5	2.0	0	4	3.0	2	35	2	2
11	5	4.0	1	10	3.0	3	3	4.0	0	11	2.0	2	11	4.0	1	5	4.0	0	45	4	2
12	3	2.0	1	5	2.0	0	5	2.0	1	7	2.0	2	12	3.0	2	5	4.0	2	37	2	2
13	1	1.0	0	5	1.0	0	4	1.0	0	8	1.0	1	13	2.0	3	6	1.0	3	37	1	1
14	5	2.0	2	9	2.0	3	6	2.0	1	14	2.0	0	14	2.0	2	6	2.0	1	54	2	2
15	2	4.5	1	6	3.0	2	6	4.0	0	11	5.0	1	7	5.0	3	5	4.0	0	37	4	1
16	3	4.0	0	6	4.0	2	4	3.0	2	10	4.0	1	9	2.0	2	4	4.0	0	36	4	2
17	1	1.0	0	4	1.5	1	1	1.0	0	3	2.0	0	7	2.0	2	7	2.0	2	23	2	2
18	2	1.0	0	4	1.0	0	3	1.0	0	5	4.0	2	4	3.0	2	5	2.0	3	23	2	3
	6			12			9			18			19			10			74		

C Vastaukset yrityksittäin

Liitteessä esitetään vastausten mediaanit yrityksittäin suuruusjärjestyksessä (Taulukot C1, C2 ja C3). Lisäksi esitetään kriittiset virhetiedot summaamalla virhetiedon hyödyllisyyden, ongelmallisuuden ja automaation helppouden mediaanit yhteen (Taulukko C4).

Taulukko C1: Vastaukset virhetietojen hyödyllisyydestä yrityksittäin.

Yritys A

Virhetieto	Mediaani
Virheen toisto-ohjeet	5
Sovelluksen osa	4.5
Havaittu käyttäytyminen	4
Sovelluksen asetukset	3.5
Käytetty data	3
Kuvakaappaukset	3
Sovelluksen komponentti	2.5
Suorituspino	2.5
Virheraportit	2.5
Tuotetiedot	2
Odotettu käyttäytyminen	2
Ohjelmistoympäristö	2
Raportoijan yhteystiedot	2
Virheen vakavuus	1.5
Käyttäjän vuorovaikutus	1.5
Virheraportin otsikko	1.5
Laitealusta	1
Testitapaukset ja -skriptit	1

Yritys B1

Virhetieto	Mediaani
Kuvakaappaukset	5
Virheen toisto-ohjeet	5
Käytetty data	5
Sovelluksen osa	5
Havaittu käyttäytyminen	5
Odotettu käyttäytyminen	4.5
Sovelluksen komponentti	4
Tuotetiedot	4
Käyttäjän vuorovaikutus	4
Virheraportin otsikko	3.5
Sovelluksen asetukset	3.5
Virheraportit	3.5
Suorituspino	3
Testitapaukset ja -skriptit	3
Virheen vakavuus	3
Ohjelmistoympäristö	2.5
Raportoijan yhteystiedot	1.5
Laitealusta	1

Yritys B2

Virhetieto	Mediaani
Sovelluksen osa	5
Virheen toisto-ohjeet	5
Suorituspino	5
Havaittu käyttäytyminen	5
Käytetty data	4
Tuotetiedot	4
Sovelluksen komponentti	4
Sovelluksen asetukset	4
Käyttäjän vuorovaikutus	4
Kuvakaappaukset	4
Virheraportit	4
Virheraportin otsikko	3
Odotettu käyttäytyminen	3
Ohjelmistoympäristö	2
Raportoijan yhteystiedot	2
Laitealusta	1
Virheen vakavuus	1
Testitapaukset ja -skriptit	1

Yritys C

Virhetieto	Mediaani
Sovelluksen osa	5
Virheen toisto-ohjeet	5
Kuvakaappaukset	4
Käyttäjän vuorovaikutus	4
Sovelluksen komponentti	4
Suorituspino	4
Havaittu käyttäytyminen	4
Sovelluksen asetukset	3.5
Käytetty data	3.5
Virheraportit	3
Odotettu käyttäytyminen	3
Ohjelmistoympäristö	2.5
Tuotetiedot	2
Virheen vakavuus	2
Raportoijan yhteystiedot	2
Virheraportin otsikko	1
Testitapaukset ja -skriptit	1
Laitealusta	1

Yritys D

Virhetieto	Mediaani
Virheen toisto-ohjeet	5
Sovelluksen osa	5
Havaittu käyttäytyminen	5
Odotettu käyttäytyminen	4
Käytetty data	4
Kuvakaappaukset	4
Sovelluksen asetukset	4
Sovelluksen komponentti	3
Virheraportit	3
Tuotetiedot	3
Testitapaukset ja -skriptit	3
Virheraportin otsikko	2
Suorituspino	2
Raportoijan yhteystiedot	2
Ohjelmistoympäristö	2
Käyttäjän vuorovaikutus	1
Virheen vakavuus	1
Laitealusta	1

Yritys E

Virhetieto	Mediaani
Virheen toisto-ohjeet	5
Käytetty data	5
Kuvakaappaukset	4
Suorituspino	4
Käyttäjän vuorovaikutus	4
Havaittu käyttäytyminen	4
Odotettu käyttäytyminen	4
Testitapaukset ja -skriptit	3.5
Sovelluksen osa	3.5
Virheraportit	3.5
Raportoijan yhteystiedot	3.5
Sovelluksen komponentti	3
Virheraportin otsikko	2.5
Virheen vakavuus	2
Sovelluksen asetukset	1.5
Ohjelmistoympäristö	1.5
Tuotetiedot	1
Laitealusta	1

Taulukko C2: Vastaukset virhetietojen puutteellisuudesta tai virheellisyydestä yrityksittäin.

Yritys A

Virhetieto	Mediaani
Laitealusta	4.5
Virheen toisto-ohjeet	3.5
Kuvakaappaukset	3.5
Sovelluksen komponentti	3
Havaittu käyttäytyminen	3
Odotettu käyttäytyminen	3
Testitapaukset ja -skriptit	3
Tuotetiedot	3
Virheraportin otsikko	3
Virheen vakavuus	3
Sovelluksen osa	3
Käytetty data	3
Raportoijan yhteystiedot	2.5
Sovelluksen asetukset	2
Ohjelmistoympäristö	2
Käyttäjän vuorovaikutus	2
Suorituspino	2
Virheraportit	2

Yritys B1

Virhetieto	Mediaani
Suorituspino	5
Testitapaukset ja -skriptit	5
Sovelluksen asetukset	4
Laitealusta	4
Odotettu käyttäytyminen	4
Ohjelmistoympäristö	4
Käyttäjän vuorovaikutus	4
Virheen toisto-ohjeet	4
Sovelluksen komponentti	3.5
Kuvakaappaukset	3.5
Käytetty data	3
Virheraportit	3
Havaittu käyttäytyminen	3
Tuotetiedot	3
Virheen vakavuus	3
Sovelluksen osa	2.5
Virheraportin otsikko	2
Raportoijan yhteystiedot	1

Yritys B2

Virhetieto	Mediaani
Ohjelmistoympäristö	4
Laitealusta	3.5
Suorituspino	3
Sovelluksen asetukset	3
Odotettu käyttäytyminen	3
Virheen toisto-ohjeet	3
Käyttäjän vuorovaikutus	3
Testitapaukset ja -skriptit	3
Käytetty data	3
Virheraportit	2.5
Virheen vakavuus	2.5
Kuvakaappaukset	2.5
Sovelluksen komponentti	2
Tuotetiedot	2
Virheraportin otsikko	2
Havaittu käyttäytyminen	2
Sovelluksen osa	2
Raportoijan yhteystiedot	1

Yritys C

Virhetieto	Mediaani
Laitealusta	5
Sovelluksen asetukset	4
Käyttäjän vuorovaikutus	4
Testitapaukset ja -skriptit	4
Virheen toisto-ohjeet	4
Ohjelmistoympäristö	3.5
Käytetty data	3.5
Kuvakaappaukset	3
Odotettu käyttäytyminen	3
Raportoijan yhteystiedot	3
Virheraportit	3
Sovelluksen osa	3
Sovelluksen komponentti	2.5
Virheen vakavuus	2.5
Virheraportin otsikko	2.5
Havaittu käyttäytyminen	2
Suorituspino	2
Tuotetiedot	2

Yritys D

Virhetieto	Mediaani
Suorituspino	5
Ohjelmistoympäristö	4.5
Sovelluksen asetukset	4
Kuvakaappaukset	4
Odotettu käyttäytyminen	4
Virheen toisto-ohjeet	3.5
Laitealusta	3.5
Käyttäjän vuorovaikutus	3
Havaittu käyttäytyminen	3
Testitapaukset ja -skriptit	3
Tuotetiedot	3
Käytetty data	3
Sovelluksen osa	3
Sovelluksen komponentti	3
Virheen vakavuus	3
Virheraportit	3
Virheraportin otsikko	2
Raportoijan yhteystiedot	2

Yritys E

Virhetieto	Mediaani
Käyttäjän vuorovaikutus	4
Odotettu käyttäytyminen	4
Sovelluksen asetukset	4
Kuvakaappaukset	4
Virheraportit	3.5
Suorituspino	3.5
Virheen toisto-ohjeet	3.5
Testitapaukset ja -skriptit	3
Tuotetiedot	3
Sovelluksen komponentti	3
Sovelluksen osa	3
Ohjelmistoympäristö	3
Laitealusta	3
Käytetty data	3
Havaittu käyttäytyminen	3
Virheen vakavuus	2.5
Virheraportin otsikko	1.5
Raportoijan yhteystiedot	1

Taulukko C3: Vastaukset virhetietojen automaation helppoudesta yrityksittäin.

Yritys A

Virhetieto	Mediaani
Laitealusta	5
Virheraportin otsikko	5
Raportoijan yhteystiedot	5
Ohjelmistoympäristö	5
Tuotetiedot	4.5
Sovelluksen komponentti	4.5
Käyttäjän vuorovaikutus	4.5
Suorituspino	4.5
Kuvakaappaukset	4
Sovelluksen asetukset	4
Käytetty data	4
Virheraportit	4
Sovelluksen osa	4
Virheen toisto-ohjeet	2
Havaittu käyttäytyminen	2
Odotettu käyttäytyminen	1
Testitapaukset ja -skriptit	1
Virheen vakavuus	1

Yritys B1

Virhetieto	Mediaani
Tuotetiedot	4.5
Laitealusta	4
Sovelluksen komponentti	4
Sovelluksen asetukset	4
Ohjelmistoympäristö	4
Virheraportit	4
Käytetty data	4
Raportoijan yhteystiedot	4
Käyttäjän vuorovaikutus	3
Suorituspino	3
Kuvakaappaukset	3
Sovelluksen osa	2
Virheen toisto-ohjeet	2
Havaittu käyttäytyminen	2
Testitapaukset ja -skriptit	1.5
Virheraportin otsikko	1
Virheen vakavuus	1
Odotettu käyttäytyminen	1

Yritys B2

Virhetieto	Mediaani
Ohjelmistoympäristö	5
Laitealusta	5
Tuotetiedot	4.5
Raportoijan yhteystiedot	4
Kuvakaappaukset	4
Suorituspino	4
Sovelluksen asetukset	4
Käytetty data	4
Virheraportit	3
Käyttäjän vuorovaikutus	2
Sovelluksen komponentti	2
Sovelluksen osa	2
Virheen toisto-ohjeet	2
Havaittu käyttäytyminen	2
Virheraportin otsikko	1
Virheen vakavuus	1
Odotettu käyttäytyminen	1
Testitapaukset ja -skriptit	1

Yritys C

Virhetieto	Mediaani
Tuotetiedot	5
Suorituspino	5
Raportoijan yhteystiedot	4.5
Sovelluksen komponentti	4
Sovelluksen osa	4
Virheraportit	4
Ohjelmistoympäristö	4
Laitealusta	4
Virheraportin otsikko	4
Virheen vakavuus	4
Käytetty data	3
Sovelluksen asetukset	2
Kuvakaappaukset	2
Havaittu käyttäytyminen	2
Käyttäjän vuorovaikutus	2
Virheen toisto-ohjeet	2
Testitapaukset ja -skriptit	2
Odotettu käyttäytyminen	1

Yritys D

Virhetieto	Mediaani
Raportoijan yhteystiedot	5
Suorituspino	5
Tuotetiedot	4.5
Ohjelmistoympäristö	4
Laitealusta	4
Kuvakaappaukset	4
Sovelluksen komponentti	4
Käytetty data	4
Sovelluksen osa	4
Sovelluksen asetukset	4
Havaittu käyttäytyminen	3
Virheen vakavuus	3
Virheraportin otsikko	2
Virheen toisto-ohjeet	2
Virheraportit	2
Testitapaukset ja -skriptit	2
Odotettu käyttäytyminen	2
Käyttäjän vuorovaikutus	2

Yritys E

Virhetieto	Mediaani
Raportoijan yhteystiedot	5
Ohjelmistoympäristö	4.5
Laitealusta	4.5
Tuotetiedot	4
Virheraportit	4
Sovelluksen asetukset	4
Käytetty data	4
Kuvakaappaukset	4
Suorituspino	4
Sovelluksen osa	4
Havaittu käyttäytyminen	4
Sovelluksen komponentti	3
Käyttäjän vuorovaikutus	3
Virheraportin otsikko	2.5
Testitapaukset ja -skriptit	2
Virheen vakavuus	2
Virheen toisto-ohjeet	2
Odotettu käyttäytyminen	1

Taulukko C4: Vastausten mediaanit summattuna yrityksittäin.

Yritys A

Virhetieto	Pisteet
Sovelluksen osa	11.5
Laitealusta	10.5
Kuvakaappaukset	10.5
Virheen toisto-ohjeet	10.5
Sovelluksen komponentti	10
Käytetty data	10
Tuotetiedot	9.5
Sovelluksen asetukset	9.5
Raportoijan yhteystiedot	9.5
Virheraportin otsikko	9.5
Ohjelmistoympäristö	9
Havaittu käyttäytyminen	9
Suorituspino	9
Virheraportit	8.5
Käyttäjän vuorovaikutus	8
Odotettu käyttäytyminen	6
Virheen vakavuus	5.5
Testitapaukset ja -skriptit	5

Yritys B1

Virhetieto	Pisteet
Käytetty data	12
Tuotetiedot	11.5
Sovelluksen komponentti	11.5
Sovelluksen asetukset	11.5
Kuvakaappaukset	11.5
Käyttäjän vuorovaikutus	11
Virheen toisto-ohjeet	11
Suorituspino	11
Ohjelmistoympäristö	10.5
Virheraportit	10.5
Havaittu käyttäytyminen	10
Sovelluksen osa	9.5
Odotettu käyttäytyminen	9.5
Testitapaukset ja -skriptit	9.5
Laitealusta	9
Virheen vakavuus	7
Raportoijan yhteystiedot	6.5
Virheraportin otsikko	6.5

Yritys B2

Virhetieto	Pisteet
Suorituspino	12
Ohjelmistoympäristö	11
Sovelluksen asetukset	11
Käytetty data	11
Tuotetiedot	10.5
Kuvakaappaukset	10.5
Virheen toisto-ohjeet	10
Laitealusta	9.5
Virheraportit	9.5
Sovelluksen osa	9
Käyttäjän vuorovaikutus	9
Havaittu käyttäytyminen	9
Sovelluksen komponentti	8
Raportoijan yhteystiedot	7
Odotettu käyttäytyminen	7
Virheraportin otsikko	6
Testitapaukset ja -skriptit	5
Virheen vakavuus	4.5

Yritys C

Virhetieto	Pisteet
Sovelluksen osa	12
Virheen toisto-ohjeet	11
Suorituspino	11
Sovelluksen komponentti	10.5
Ohjelmistoympäristö	10
Laitealusta	10
Käytetty data	10
Käyttäjän vuorovaikutus	10
Virheraportit	10
Sovelluksen asetukset	9.5
Raportoijan yhteystiedot	9.5
Tuotetiedot	9
Kuvakaappaukset	9
Virheen vakavuus	8.5
Havaittu käyttäytyminen	8
Virheraportin otsikko	7.5
Odotettu käyttäytyminen	7
Testitapaukset ja -skriptit	7

Yritys D

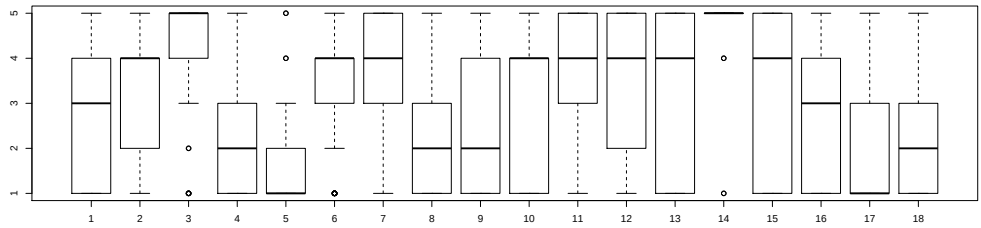
Virhetieto	Pisteet
Sovelluksen osa	12
Sovelluksen asetukset	12
Kuvakaappaukset	12
Suorituspino	12
Käytetty data	11
Havaittu käyttäytyminen	11
Tuotetiedot	10.5
Ohjelmistoympäristö	10.5
Virheen toisto-ohjeet	10.5
Sovelluksen komponentti	10
Odotettu käyttäytyminen	10
Raportoijan yhteystiedot	9
Laitealusta	8.5
Virheraportit	8
Testitapaukset ja -skriptit	8
Virheen vakavuus	7
Virheraportin otsikko	6
Käyttäjän vuorovaikutus	6

Yritys E

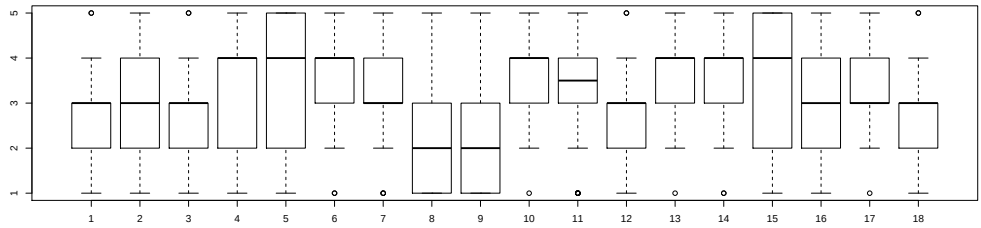
Virhetieto	Pisteet
Käytetty data	12
Kuvakaappaukset	12
Suorituspino	11.5
Käyttäjän vuorovaikutus	11
Havaittu käyttäytyminen	11
Virheraportit	11
Sovelluksen osa	10.5
Virheen toisto-ohjeet	10.5
Sovelluksen asetukset	9.5
Raportoijan yhteystiedot	9.5
Sovelluksen komponentti	9
Ohjelmistoympäristö	9
Odotettu käyttäytyminen	9
Laitealusta	8.5
Testitapaukset ja -skriptit	8.5
Tuotetiedot	8
Virheraportin otsikko	6.5
Virheen vakavuus	6.5

D Laatikkokuviot vastauksista

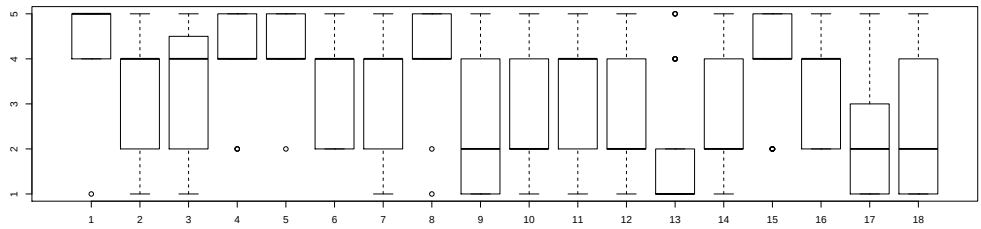
Liitteessä esitetään vastausten hajontaa havainnollistavat laatikkokuviot, joissa vastausten mediaani esitetään paksuna viivana ja kvartiiliväli laatikkona (Kuva D1).



1. ja 2. kysymyksen vastaukset: hyödyllisyys



3. kysymyksen vastaukset: puutteellisuus tai virheellisyys



4. kysymyksen vastaukset: automaattisen keräämisen helppous

1. Tuotetiedot
2. Sovelluksen komponentti
3. Sovelluksen osa
4. Ohjelmistoympäristö
5. Laitealusta
6. Sovelluksen asetukset
7. Käytetty data
8. Raportoijan yhteystiedot
9. Virheraportin otsikko
10. Käyttäjän vuorovaikutus
11. Kuvakaappaukset
12. Havaittu käyttäytyminen
13. Odotettu käyttäytyminen
14. Virheen toisto-ohjeet
15. Suorituspino
16. Virheraportit
17. Testitapaukset ja -skriptit
18. Virheen vakavuus

Kuva D1: Kyselyn vastausten hajonnan havainnollistaminen laatikkokuvioina.

E Kyselyssä vastatut virhetiedot

Kyselyn vastausten perusteella seuraavat virhetiedot ovat hyödyllisiä virheiden korjauksessa:

1. Information on behavior of multiple users
2. Video, how to reproduce
3. log files
4. Video capture of how to reproduce (which shows also the ”unlisted steps”)
5. Network configuration between server and client.
6. URL of the page in web application
7. Oracle trace files
8. Credentials (user account, for example, admin, power user) used when the bug occurred
9. SQL trace files
10. How much reporter has used the system (how familiar reporter is with the system).
11. test site url, server name
12. Dump of the Telecommunication
13. Information on user’s (tester’s) context eg. his/her usergroup, privileges, etc.
14. Software/product version where the bug was found
15. Info if has been working sometimes, or always been broken. If has been working, when?
16. database information (ip address, username, password)
17. Anything other unusual, e.g., Other program crashes, network problems...
18. Data from Session state of the application
19. Selected database query results
20. public site url, server name
21. Was it a new bug, or something that never worked?
22. caseid