

Opinnäytetyö (AMK)

Tietojenkäsittely

Tietojärjestelmät

2011

Toni Nummela & Atso Hiltunen

MIKROTUKIPALVELUN HUOL- TOTIETOKANTA



TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

Toni Nummela & Atso Hiltunen

MIKROTUKIPALVELUN HUOLTOTIETOKANTA

Kansalaisen mikrotuki on Turun ammattikorkeakoulun oppimisympäristö, jonka tarkoituksena on tarjota maksutonta apua ja koulutusta tietokoneasioissa. Kansalaisen mikrotuessa haluttiin vähentää paperin käyttöä. Tätä varten tarvittiin uusi järjestelmä, johon tallennettaisiin aikaisemmin paperilla ollutta tietoa. Samalla todettiin, että irrallaan olevia tietoja voisi yhdistää yhden järjestelmän alle.

Projekti toteutettiin käyttämällä Extreme Programmingia, joka on yksi ketterän kehityksen menetelmistä. Extreme Programmingissa ohjelmointi tapahtuu pareittain saman koneen äärellä, jolloin kaikkiin ratkaisuihin saa välittömän palautteen. Kaikissa ketterän kehityksen menetelmissä yhteistyö asiakkaan kanssa on tärkeää. Projektin edetessä ilmeni tarvetta uusille ominaisuuksille, joita ei perinteisillä ohjelmistonkehitysmenetelmillä olisi edes harkittu lisättäväksi.

Käyttöliittymää tehdessä oli tärkeää luoda oma standardi, jonka mukaisesti kaikki lomakkeet tehtiin. Oman standardin mukaisesti tehty käyttöliittymä on joka tilanteessa yhteneväinen eikä käyttäjälle tuota ongelmia tilanteesta toiseen siirtyessä. Käyttäjät ovat myös tottuneet tietynlaisiin käyttöliittymiin, joiden perusteet on luotu yli 25 vuotta sitten. Tämä otettiin huomioon standardia luodessa.

Ohjelmistoa ei voi ottaa käyttöön ilman kattavaa testausta. Jokainen ohjelmaa käyttäessä ilmenevä virhe heikentää työtehoa, kun asia pitää saada tehtyä, mutta normaalein keinoin se ei onnistu. Testauksessa tärkeintä on käydä mahdollisia ongelman aiheuttajia läpi järjestelmällisesti, jotta mahdollisimman moni tilanne tutkittaisiin. Sattumanvaraisesti syötteiden kokeilemista kutsutaan joskus testaukseksi, mutta sillä vain yritetään todistaa ohjelmiston toimintaa eikä etsiä virheitä.

Järjestelmään voi käyttää vain Kansalaisen mikrotuen sisäverkosta. Ulkoiset riskit tiedonsaannin osalta hoituvat tällä. Järjestelmää käytetään vain huoltotilassa, jolloin asiakkaat eivät pääse näkemään muiden tietoja. Sisäverkkoa käytetään myös joskus asiakastilassa, kun tietokonetta huolletaan asiakkaan odottaessa. Tällöin työntekijät eivät saa jättää konetta valvomatta, sillä näissä tilanteissa järjestelmään voi päästä sisään.

ASIASANAT:

Extreme Programming, ketterä kehitys, käyttöliittymät, ohjelmistokehitys, testaus, tietoturva

BACHELOR'S THESIS | ABSTRACT
TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Information Systems

August 2011 | 57 pages

Instructor Anne Jumppanen

Toni Nummela & Atso Hiltunen

SERVICE DATABASE OF A HELPDESK SERVICE

Citizen's Helpdesk is a learning environment at Turku University of Applied Sciences. Its purpose is to offer free help and training concerning computer issues. Citizen's Helpdesk wanted to reduce paper usage. A new database system was needed to save the information that was written on paper at the time. Several separate systems could be combined while creating this new system.

The project was implemented using Extreme Programming, which is a agile development method. Programming in Extreme Programming is done in pairs so that you get instant feedback about everything. Like in any agile method, client collaboration is important. During the implementation, a need arose for new features. They couldn't be implemented while using a traditional method.

Creating our own standard was important when designing the user interface. The use of a standard when creating the user interface makes it easier for the user to navigate in the system. There are issues the users are familiar with in other user interfaces. They were documented over 25 years ago and had to be taken into account of while designing the standard.

A program has to be tested thoroughly before using it. Every error in the system decreases efficiency as you have to get things done but have to find your way around instead of doing it by normal means. The most important thing in testing is to be systematic to cover as many possible situations as possible. Testing using random inputs just tries to prove that the program works instead of trying to find bugs.

Access to the system is only possible from the network in Citizen's Helpdesk. This way external risks of data leaking are taken care of. Clients can't see information about other clients as the system is only used on maintenance side. Sometimes the network is also used on client side to fix things while the client is waiting. Employees shouldn't leave computers plugged into network unattended to make sure the system stays uncompromised.

KEYWORDS:

agile development, Extreme Programming, information security, software development, software testing, user interface

SISÄLTÖ

| | |
|------------------------------------|-----------|
| 1 JOHDANTO | 6 |
| 2 NYKYTILA JA TAVOITETILA | 8 |
| 2.1 Lähtökohta | 8 |
| 2.2 Tavoitetila | 10 |
| 2.3 Uusi järjestelmä | 10 |
| 3 KETTERÄ OHJELMISTOKEHITYS | 12 |
| 3.1 Agile manifesto | 13 |
| 3.2 Tiimityö | 14 |
| 3.3 Extreme programming | 17 |
| 4 KÄYTTÖLIITTYMÄT | 20 |
| 4.1 Periaatteet | 21 |
| 4.2 Rakenne | 23 |
| 4.3 Yleisilme | 29 |
| 5 TESTAUS | 31 |
| 6 TIETOTURVA | 35 |
| 6.1 Tietoturvapoliitikat | 35 |
| 6.2 Tietoturva KMT:llä | 37 |
| 6.3 Etäseuranta | 38 |
| 7 JOHTOPÄÄTÖKSET | 40 |
| LÄHTEET | 42 |

LIITTEET

- Liite 1. Tietokannan rakenne
- Liite 2. Järjestelmän käyttöopas

KUVAT

| | |
|--|----|
| Kuva 1. Vanhan järjestelmän vastaanottolomake | 7 |
| Kuva 2. KMT:n asiakastila. Koneiden vastaanotto vasemmalla ja opastukset oikealla. | 9 |
| Kuva 3. Käyttöjärjestelmän valinta | 24 |
| Kuva 4. Tekstilaatikat | 27 |

| | |
|---|----|
| Kuva 5. Vastaanottolomake | 28 |
| Kuva 6. Etusivu | 29 |
| Kuva 7. Kansalaisen mikrotuen vanha järjestelmä | 30 |

KUVIOT

| | |
|--|----|
| Kuvio 1. Testauksen V-malli (Haikala & Märijärvi 2000, 271). | 31 |
|--|----|

1 JOHDANTO

Kansalaisen mikrotuki on Turun ammattikorkeakoulun oppimisympäristö, jonka tarkoituksena on tarjota maksutonta apua ja koulutusta tietokoneasioissa. Työ oli toimeksianto Kansalaisen mikrotuen johtavalta projektipäälliköltä. Tarkoitus oli tehostaa KMT:n toimintaa yhdistämällä järjestelmiä ja vähentää paperin käyttöä kirjaamalla kaikki tarpeellinen tieto suoraan järjestelmään.

Molemmat tekijät suorittivat harjoittelunsa KMT:llä. Suunnittelun lähtökohdiksi otettiin harjoittelun aikana huomattujen ongelmien korjaaminen. Projektissa käytettiin mahdollisimman paljon ketterän kehityksen menetelmiä. Metodiksi valittiin pariohjelmoinnin ja oman kiinnostuksen vuoksi Extreme Programming.

Työn tekeminen aloitettiin tehtävän järjestelmän vaatimusten kirjaamisella. Asiakkaan vaatimuksia olivat koneiden kirjaaminen tietokantaan, asiakastapahtumien kirjaus, työtuntien kirjaus, ajanvaraus ja tilanneseuranta netin välityksellä, jos KMT:n verkkoon saadaan yhteys ulkopuolelta. Ulkopuolelta yhteyden ottaminen KMT:n verkkoon ei onnistu, joten tilanneseurantaa ei järjestelmään tehty.

Näiden perusteella suunniteltiin tietokanta, jonka pohjalle järjestelmän voisi rakentaa. Tietokantaa muutettiin järjestelmän kehityksen edetessä vastaamaan muuttuvia vaatimuksia. Tietokannan kuvaus liitteenä 1.

Käyttöliittymän suunnittelussa lähdettiin liikkeelle vanhan järjestelmän ulkoasusta (kuva 1) ja pidettiin pääelementit samoilla alueilla, jotta vanhojen työntekijöiden siirtyminen uuteen järjestelmään olisi mahdollisimman helppoa.

Kansalaisen Mikrotuki
Lekalla siitä selvää

Haku:

Valikko

- Kirjaudu ulos
- Etusivu
- Varmuuskopiointi
- Asiakashallinta
- Lisää tietokone
- Lisää digiboksi
- Haku
- Asetukset
- Ohjeet

Kansalaisen Mikrotuki
Lisää asiakas

Nimi * :

Puhelin * :

Saapunut:

Valmistunut:

Vikakuvaus:

Salasana:

Toimenpiteet:

Vastaanottaja * :

Käsitelijät:

Tuntomerkit:

Tietoturva:

Käyttöjärjestelmä:

Verkko:

Copyright Kari Mattila, 2007

Kuva 1. Vanhan järjestelmän vastaanottolomake

Opinnäytetyön tavoitteena on tutustua ketterään kehitykseen, tarkemmin Extreme Programmingiin, käyttöliittymien suunnitteluun ja tietojärjestelmien tietoturvaan.

Työnjakona Atso kirjoitti käyttöliittymiin ja tietoturvaan liittyvät luvut, Toni ketterään ohjelmistokehitykseen ja testaukseen liittyvät luvut. Empiirisen osan ja muut luvut teimme yhdessä.

2 NYKYTILA JA TAVOITETILA

2.1 Lähtökohta

Kansalaisen mikrotuki (KMT) on Turun ammattikorkeakoulun oppimisympäristö. KMT:n tarkoituksena on auttaa turkulaisia tietokoneongelmien kanssa ja antaa opiskelijoille arvokasta työkokemusta atk-laitteista, asiakaspalvelusta ja johtamisesta. KMT:ssä työskentelevät Turun ammattikorkeakoulun opiskelijat projektipäällikkönä ja tämän apulaisena.

Näiden lisäksi Turun ammattikorkeakoulun opiskelijat suorittavat kursseja KMT:ssä, ja Turun ammatti-instituutin ja Kupittaaan IT-lukion opiskelijoita käy suorittamassa työssäoppimis- ja TET-jaksoja. Täysin opiskelijavoimalla KMT ei kuitenkaan pyöri, sillä toimintaa johtaa Turun AMK:n lehtori Virpi Raivonen. Tässä työssä KMT:n työntekijöillä tarkoitetaan kaikkia edellä mainittuja.

KMT on ilmainen palvelu ihmisille, jotka tarvitsevat apua atk-laitteidensa kanssa. KMT:ssa hoidetaan kaikenlaisia ongelmia. Laitevikojen kanssa ohjataan alan liikkeeseen, koska resursseja osien vaihtoon ja tarkkaan analyysiin ei ole. Lisäksi mikäli koneelle sattuisi korjattaessa jotain, ei olisi varaa korvata asiakkaalle menetyksiä.

Yleisiä ongelmia ovat virusongelmat, koneen hitaus ja osaamattomuus atk-asioissa. KMT:llä on myös puhelin, johon voi soittaa ja kysyä neuvoa. Mikäli puhelimesta ei pystytä ratkaisemaan ongelmaa, kutsutaan asiakas käymään toimipisteessä.

KMT on jaettu kahteen osaan, joita ovat asiakastila ja huoltotila. Tämä toimii hyvin, sillä tällä taataan työntekijöille mahdollisimman hyvä työrauha ja asiakkaat eivät näe toisten asiakkaiden tietoja tai tiedostoja. Mikäli kyseessä on opastus, se hoidetaan asiakastilassa (kuva 2), jolla on useampi pöytä ja pistorasiala asiakkaiden koneita varten.



Kuva 2. KMT:n asiakastila. Koneiden vastaanotto vasemmalla ja opastukset oikealla.

KMT:ssä tieto on jakautunut eri paikkoihin ja eri järjestelmiin. On vikakuvakauslomakkeita, asiakastapahtumien Excel-taulu, työntekijöiden työtunnit useammassa Excel-tiedostossa ja valmiit koneet -tietokanta. Koneen tiedot kirjoitetaan tietokantaan vasta, kun kone on valmis, joten ainoa tapa koneiden tunnistamiseen ovat laput koneiden päällä.

Koneista tulee usein kyselyjä, joissa halutaan tietää huollon tila. Tiedon paikallistaminen on välillä vaikeaa, koska täytyy lähteä etsimään lappuja, jotka ovat koneiden päällä ennalta määräämättömässä paikassa huoltotilassa.

Työntekijät pitävät kirjaa tunneistaan ja siitä, mitä ovat suoritetuilla tunneilla tehneet ja oppineet. Tämän kirjanpidon pohjalta laaditaan kurssin suoritukseen vaadittava loppuraportti. Asiakastapahtumista ja puheluista täytyy pitää kirjaa, jotta tiedetään kuinka paljon asiakkaita käy ja koneista täytyy olla selvillä, mikä

on ollut vikana, mitä sille on tehty ja kuka on huoltanut. Koska opiskelijat suorittavat satunnaisen tiheästi KMT-kurssia vuosien aikana, olisi hyvä jos pystyisi yhdistämään tehdyt tunnit ja koneet. Usein opiskelijat kirjaavat ylös vain "huolisin konetta - 2H", mutta mitä konetta on huollettu ja mikä siinä on ollut ongelma jäävät pois. Se, mitä on tehnyt esimerkiksi kolme vuotta sitten koneen korjaukseksi, voi olla jo aikoja sitten unohtunut.

2.2 Tavoitetila

Tavoitteena on, että koska kaikki tiedot ovat yhdessä paikassa, ei tarvitse etsiä aina oikeaa tiedostoa, kun merkitsee asioita muistiin. KMT:n loppuraporttia kirjoittaessa ei tarvitse enää miettiä mitä on tehnyt, sillä tuntikirjaukset ovat sidottuina koneisiin joista näkee helposti mitä niille on tehty.

Järjestelmässä kirjataan kaikki koneiden tiedot suoraan muistiin, jolloin ei tarvitse enää kopioida vastaanottolomakkeita vaan tulostetaan valmiiksi kirjoitettu lomake suoraan. Tämä vähentää käsin kirjoitetun tekstin määrää huomattavasti, jolloin ihmisten ei tarvitse tulkata toistensa käsialaa.

2.3 Uusi järjestelmä

Tarkoituksena uudella järjestelmällä on yhdistää erillään olevat tiedot yhteen paikkaan helposti saataville. Lisäksi järjestelmä tekee muutamia asioita itse, kuten yhdistää tehdyt tunnit työntekijöihin ja työntekijät koneisiin, jolloin ihmisten ei tarvitse pitää erillistä kirjanpitoa tekemistään toimenpiteistä ja he näkevät jälkikäteen helposti, mitä koneille on tehty.

Järjestelmä antaa myös tehdä vastaanottolomakkeen koneella, jolloin ei tarvitse enää tulkata toisten käsialaa ja miettiä, että onkohan puhelinnumero nyt oikein. Järjestelmän tehtävänä on myös auttaa laskemaan, kuinka monta asiakastapahtumaa on tullut milläkin ajanjaksolla, mikä puolestaan helpottaa projektipäällikön tehtävää.

Paperin käytön määrää emme voineet valitettavasti vähentää uudella järjestelmällä, sillä KMT:ssa tarvitaan kuitenkin allekirjoitettava vastaanottolomake ja

lopuksi asiakkaalle annetaan huoltolomake, jossa kerrotaan, mitä koneelle on tehty ja mahdollisesti annetaan ohjeita siitä, miten konetta voisi käyttää siten, ettei vastaavia ongelmia tulisi uudelleen.

3 KETTERÄ OHJELMISTOKEHITYS

Perinteisesti ohjelmistonkehitys on ollut operaatio, joka alkaa tulevan ohjelmiston tarkasta määrittelystä ja päättyy ylläpitovaiheeseen. Tämän operaation esittävää mallia kutsutaan vesiputousmalliksi. Vesiputousmallin eri vaiheet ovat prosesseja, jotka yrityksissä määritellään tarkasti. Tällöin jokainen prosessissa mukana oleva tekee asiat tietyllä tavalla ja dokumentoinnilla, eikä henkilöiden vaihtuminen näytä vaikuttavan projektin etenemiseen suuresti.

Vesiputousmalli toimii huonosti tilanteessa, jossa vaatimuksista ei olla täysin varmoja tai ne suurella todennäköisyydellä muuttuvat projektin aikana. Näitä tilanteita varten tarvitaan toisenlainen toimintatapa. Tällaiseksi on muodostunut ketterän kehityksen peruseriaate ja tästä on useita erilaisia ketteriä kehitysmalleja. Näillä malleilla on erilaisia toimintatapoja, mutta kaikki perustuvat Agile manifestoon, jota käsitellään luvussa 3.1.

Perinteisesti ohjelmoinnissa menestys on tullut tuntien työskentelyllä yksin, puhumatta kenenkään kanssa ja tutkimalla papereita ja ruutua. Tähän soveltumattomat ihmiset ovatkin lähteneet alalta. Uudemmat, varsinkin ketterät kehitysmetodit kuitenkin painottavat kommunikointia. Yllättäen henkilöiltä, jotka ovat hakeutuneet vähän kommunikaation alalle, vaaditaan hyvää kommunikaatio-osaamista. (Cockburn 2007, 37-38.)

Yleisesti ohjelmoijien ajatellaan olevan epäsosiaalisia ihmisiä. He kuitenkin keskustelevat mielellään teknisistä asioista. Jutustelu epäolennaisista asioista, kuten urheilusta tai syntymäpäiväjuhlista, ei ohjelmoijia taas kiinnosta. Työnteon keskeyttämistä he inhoavat eniten. Tähän on kuitenkin hyvä syy. Ohjelmointi vaatii monimutkaisten ajatusketjujen yhdistämistä, johon menee paljon aikaa. Jos tämä prosessi katkaistaan, koko ajatusketjun palauttamiseen saattaa mennä 20 minuuttia ja eteenpäin pääsemiseen jopa tunti. Tämän takia enemmän kuin pari minuuttia kestävä puhelu, keskustelu tai palaveri saattaa tuhlata tunnin

työaikaa ja hyvin paljon energiaa. Epäsosiaalinen käytös on ammatillinen puolustuskeino tätä vastaan. (Cockburn 2007, 138.)

3.1 Agile manifesto

Agile manifesto painottaa henkilöitä, vuorovaikutusta ja toimivaa ohjelmistoa prosessien, työkalujen ja kattavan dokumentaation sijaan. Tämän lisäksi asiakasyhteistyö ja muutoksiin vastaaminen ovat tärkeämpiä kuin sopimusneuvottelut ja suunnitelman seuraaminen. Kaikki asiat ovat tärkeitä, mutta perinteistä poiketen tärkeysjärjestys on erilainen. Agile manifesto perustuu kahteentoista periaatteeseen:

- Tärkein prioriteetti on asiakkaan tyytyväisyyden takaaminen nopealla ja jatkuvalla ohjelmiston toimituksella.
- Suhtaudu myönteisesti muuttuviin vaatimuksiin, jopa kehityksen myöhäisessä vaiheessa. Ketterät prosessit valjastavat muutokset asiakkaan hyödyksi.
- Toimita toimivaa ohjelmistoa usein, muutamista viikoista muutamien kuukausien välein.
- Liikemiesten ja ohjelmistokehittäjien tulee toimia yhdessä päivittäin projektin läpi.
- Rakenna projektit motivoituneiden henkilöiden ympärille. Anna heille sopiva ympäristö ja tuki ja luota työn tulevan tehdyksi.
- Kehitysryhmälle ja ryhmän sisällä informaation kuljettaminen onnistuu tehokkaimmin kasvokkain.
- Toimiva ohjelmisto on ensisijainen etenemisen mittari.
- Ketterät prosessit tukevat jatkuvaa ohjelmistokehitystä. Rahoittajien, kehittäjien ja käyttäjien olisi voitava ylläpitää tasaista tahtia jatkuvasti.

- Jatkuva huomion kiinnittäminen tekniseen osaamiseen ja hyvä suunnitelu parantaa ketteryyttä.
- Yksinkertaisuus on tärkeää.
- Parhaat arkkitehtuurit, vaatimukset ja suunnitelmat tulevat itseorganisoidulta tiimiltä.
- Tiimin tulisi tasaisin väliajoin miettiä, miten se tulisi tehokkaammaksi ja muuttaa toimintaansa tämän perusteella. (Agile Alliance 2001.)

3.2 Tiimityö

Pääpainot ketterän kehityksen menetelmissä ovat tiimin sisäisessä ja ulkoisessa kommunikaatiossa. Ulkoisen kommunikaation pääpaino on tiimin ja asiakkaan välillä. Asiakkaan puolelta on tärkeää priorisoida järjestelmän kehitys korkealle ja keskittää siihen tarpeeksi voimavaroja. Kommunikaatio vaikeutuu huomattavasti jos asiakkaalla ei ole tarpeeksi osaamista ohjelmistokehityksestä, joten näiden menetelmien käyttöä tulee harkita tapauskohtaisesti.

Tiimit olisivat usein valmiita käyttämään ketteriä menetelmiä, mutta johtajat pelkäävät menettävänsä kontrollin projekteista. Dokumentaation puuttuessa tiimin sisäistä kommunikaatiota voi olla vaikea seurata. Tiimin toiminnan mittarina tulisi kuitenkin pitää tuotettua ohjelmistoa. Ensisijaisen tavoitteen eli valmiin ohjelmiston jälkeen työstä tulisivat tehdä tarpeellinen dokumentaatio siltä varalta, että tiimin rakenne muuttuu tai jäsenet tekevät välissä muita projekteja ja palaavat ohjelmiston pariin unohdettuaan ajatuksensa projektin aikana. Dokumentoinnin tekoa voisi ajatella projektiin kuuluvana aliprojektina, joka syö resursseja pääprojektilta ja joka viimeistellään sen loputtua. (Kalermo & Rissanen 2002, 84-86; Cockburn 2007, 41-45.)

Pienet tiimit toimivat suuria tehokkaammin. Täydellisessä tilanteessa tiimi on pieni ja sisältää vain kokeneita kehittäjiä. Kukaan ei kuitenkaan aloita kokeneena, joten todellisuudessa tiimit usein sisältävät aloittelevia kehittäjiä, joita kokenempien tulee ohjastaa projektin etenemiseen. Aloittelevat kehittäjät pysyvät

tiukasti oppimisissaan toimintatavoissa. Kokemuksen myötä osa opituista tavoista poistuu ja lopulta jokainen työskentelee omalla, itselleen parhaiten sopivalla tyyllillään. (Cockburn 2007, 14-17, 224.) Tiimityöskentelyssä risteävät tyylit saatavat kuitenkin sotkea asioita, joten varsinkin muodostuvasta dokumentoinnista ja koodin kommentointitavoista tulisi olla tiimin, tai jopa yrityksen, sisäiset standardit.

Täysin automaattisten testaustyökalujen avulla voi tarkistaa muokatun koodin toimivuuden napin painalluksella. Tällöin kömpelöt moduulit voidaan korvata tai niitä parantaa tuottamatta uusia virheitä. Kehittäjät voivat myös rentoutua viikonloppuna paremmin tietäen, että testi kertoo jos järjestelmää on muutettu heidän tietämättään. Kaikkea ei tietenkään voi testata automaattisesti, mutta kokeneet kehittäjät minimoivat tällaisten osien määrän ja mahdollisesti kehittävät automaattisia testejä omille moduuleilleen. (Cockburn 2007, 224.)

Hyviin ohjelmointitapoihin kuuluu ratkaisujen tekeminen muotoon, jossa niitä voidaan hyödyntää tulevaisuudessa. Amerikkalaiset ja eurooppalaiset kehittäjät eivät kuitenkaan yleensä tutki vanhoja ratkaisuja ja muokkaa näitä projektiin sopiviksi. Sen sijaan he vain kehittävät uuden ratkaisun joka kerta. Tämä juontaa juurensa koulunkäyntiin. Koulun alusta asti muiden ideoiden kopioimisesta rangaistaan ja kavereiden auttamista paheksutaan, jollei jopa kielletä. Vielä yliopistotasollakin tehtävät suunnitellaan niin, että näistä voi antaa yksilölliset arvosanat tiimityön arvosanan sijaan. (Cockburn 2007, 74-75.)

Tiimin koon ja kokemuksen lisäksi sen psykologinen rakenne vaikuttaa tehokkuuteen. MIT:n tutkijoiden mukaan ryhmä käyttää yhdessä älyään paremmin, jos se koostuu toisten tunteille herkistä yksilöistä. Naiset ovat keskimäärin miehiä parempia tässä. Stanfordin yliopiston tutkijoiden mukaan puoliksi narsisteista koostuva ryhmä taas saa uusia ideoita. Narsistit eivät keksi muita parempia ideoita, mutta uskovat ideoihinsa eniten ja innostavat muita. (Tehokkain työryhmä: valtaosa naisia, puolet narsisteja 2010.)

Tiimin tulisi yleensä sisältää vähintään yksi käyttöliittymäasiantuntija. Tällöin käyttöliittymän muutosehdotuksiin voidaan reagoida nopeammin. Asiantuntijan

puuttuessa käyttäjäpalautteen saaminen vaikeutuu ja saattaa unohtua kokonaan. Näin ei kuitenkaan saisi tapahtua, vaan sen saamisen eteen tulisi tehdä töitä. (Cockburn 2007, 223.)

Parhaassa tapauksessa tiimin työtavat syntyvät itsestään ajan myötä, mutta tällöin siitä ei jää kirjallista dokumentaatiota. Jäsenillä on päässään paljon projektiin liittyvää, paperille dokumentoimatonta tietoa ja uuden henkilön tuominen tiimiin vaikeutuu. Tiimin strategian tulee sisältää tiedon siirtäminen tuleville jäsenille mahdollisimman hyvin. Hyvin tehdystä tiimistä kehittyy ajan myötä yhteinen näinen, jolloin jäsenet luottavat toisiinsa. Tällaiset tiimit ovat esimiehille vaikeita, sillä ne sisältävät usein lempinimiä ja sisäpiirijuttuja, jotka eivät aukea ulkopuolisille. Tämän vuoksi tällaiset tiimit usein hajotetaankin. Tätä ei kuitenkaan tulisi tehdä, sillä uuteen tiimiin sisään pääsy on vaikeaa eikä samaan kokonaistehokkuuteen välttämättä pääse pitkänkään ajan päästä. (Cockburn 2007, 49-50; Kallermo & Rissanen 2002, 2-9, 71-73.)

Kuten aikaisemmin todettiin, kommunikaatio on ketterissä metodeissa ensisijaisen tärkeää. Tieto kulkee tehokkaimmin, kun samassa huoneessa työskentelee kahdesta kahdeksaan ihmistä. Tällaisessa tilanteessa voi vastata kysymyksiin ja kuulla tärkeää tietoa katkaisematta työntekoa. Tiedon kulkemisen tehokkuus laskee jokaisen oven, kulman ja hissien myötä. Myös pari-ohjelmointi, jossa kaksi henkilöä on saman ruudun äärellä tekemässä ohjelmointia, edistää kommunikointia. Standardien ylläpito ja testitapausten tekeminen on helpompaa, kun kaksi henkilöä korjaa toisiaan koko ajan. Tähän kuitenkin vaaditaan tottumisjakso, jonka aikana toisen henkilön oikut tulevat tutuiksi. (Cockburn 2007, 78, 222.)

Vaikka kommunikaatio on tärkeää, ei ihmisten tulisi istua vierekkäin koko ajan. Kuten šakissa on erilaiset strategiat alku-, keski- ja loppupelissä, myös projektien eri vaiheissa tulisi toimia hieman eri tavalla. Joskus ihmiset haluavat olla rauhassa ohjelmoimassa tai tutkimassa asioita, joskus taas tarvitaan kommunikointimodeja tilanteisiin, joissa kaikki eivät ole paikalla. Rauhaa tarvitsevat ihmiset, kuten suunnittelijat projektin seuraavaa vaihetta suunnitellessaan, kannattaa sijoittaa etäälle muista. Jo kymmenen metriä on tarpeeksi etäisyyttä vähentämään häiritsevien kysymysten määrää huomattavasti. Oikeanlainen tasa-

paino projektin eri vaiheissa on tärkeää aikataulun pitämisen kannalta. (Cockburn 2007, 101-102.)

Työryhmien tulisi pitää projektipalavereja muutaman viikon välein. Pelkkä loppupalaverin pitäminen ei auta projektin etenemisessä. Tasaisin väliajoin pidettävät tarkastelupalaverit antaisivat toimijoille palautetta ajoissa ongelmien korjaamiseksi. Tiimin olisi hyvä viettää ainakin tunti viikossa yhdessä tutkimassa työskentelyään. (Cockburn 2007, 90-91, 228.)

Nopea palautteen saaminen sekä ohjelmistosta että kehitysprosessista on erityisen tärkeää. 1-3 kuukauden välein toimitettavat ohjelmistoversiot helpottavat tämän palautteen saamista ja prosessien korjaamista. Pitkillä väliajoilla ihmisten keskittyminen projektiin laskee, mutta tämän lisäksi uuden version käyttöönoton hinta vaikuttaa sopivan aikavälin valitsemiseen. Jos ryhmä ei pysty muutaman kuukauden välein valmistamaan uutta versiota, tulisi julkaisuun liittyvät prosessit kuitenkin tehdä. Tällöin kaikkiin kehitysprosessin vaiheisiin saadaan harjoitusta ja mahdolliset julkaisuversion tekoon liittyvät ongelmat voidaan ratkaista. (Cockburn 2007, 223.)

3.3 Extreme programming

Extreme programming on pienille, noin 3-10 hengen tiimeille suunniteltu ketterän kehityksen menetelmä. Tarkoituksena on tehdä lyhyissä muutaman viikon jaksoissa toimivaa ja testattua koodia. Asiakkaalle ohjelmasta toimitetaan toimiva versio kahdesta viiteen jakson välein. (Cockburn 2007, 199.)

Extreme programmingilla tehtävässä projektissa asiakkaat, tässä tapauksessa KMT:n työntekijät, kirjoittavat omalla terminologiallaan vaatimuksia järjestelmän toimintoihin. Jokaisesta toiminnosta kirjoitetaan noin kolme lausetta. Suurin ero normaaliin vaatimusmäärittelyyn on tarkkuus. Näiden vaatimusten perusteella tehdään arvio kehityksen kestosta. (Wells 2009; Brewer 2001.) Tässä projektissa kirjallisten vaatimusten sijaan ominaisuuksien vaatimukset käsiteltiin keskustelemalla implementaatiohetkellä Kansalaisen mikrotuen projektipäällikkönä olleen henkilön kanssa.

Toimintojen kehityksen kestoarvioiden perusteella tehdään julkaisusuunnitelma, jonka tulisi olla mahdollisimman tarkka arvio projektin etenemisestä. Kehittäjien, asiakkaiden ja johtajien tulisi päästä yhteisymmärrykseen suunnitelmasta ja neuvottelujen jatkua kunnes tämä saavutetaan. Jokaisen jakson alussa pidetään palaveri, jossa kehittäjät ja asiakkaat sopivat jakson aikana kehitettävistä toiminnoista ja käyvät läpi näiden yksityiskohdat. (Wells 2009, Brewer 2001.) Tässä projektissa yhden toiminnon kehittämiseen ja testaamiseen meni noin viikko. Tarkoista ominaisuuksista oli palaveri aina, kun uutta toimintoa tarvittiin. Tämän vuoksi kehityksessä oli välillä useammankin viikon taukoja.

Kehitys alkaa testien tekemisellä. Hyvin tehdyt testit säästävät paljon aikaa tulevaisuudessa. Kaikki koodi testataan, jotta ohjelmistossa olisi mahdollisimman vähän virheitä. (Wells 2009; Brewer 2001.) Tässä projektissa normaalista poiketen ei käytetty automaattisia testejä, vaan lomakkeet testattiin sekä ennalta mietityillä syötteillä että sattumanvaraisilla syötteillä.

Kaikki kehitystyö tapahtuu pareittain. Ihmisiä on hyvä siirrellä ja pareja vaihdella, jotta kaikilla kehittäjillä olisi kokonaiskuva ohjelmistosta. Koodi tulisi pitää mahdollisimman yksinkertaisena ja tarvittaessa pari voi kirjoittaa osion uudelleen, jos se vaikuttaa liian monimutkaiselta. (Wells 2009.) Tämän projektin aikana uudelleenkirjoitettiin osioita kolme kertaa. Tällä tavalla yksinkertaistettiin osia, joihin oli lisätty uusien toimintojen takia tarkistuksia tai vaihtoehtoja.

Aina sopivan tauon tullessa ja testien mennessä läpi tulisi koodi integroida järjestelmään. Lyhyet integrointivälit, enintään yksi viikko, vähentävät yhteensopivuusongelmia, jotka saattaisivat venyttää projektin valmistumista viikoilla. (Wells 2009; Brewer 2001.) Tässä projektissa jokainen uusi toiminto integroitiin välittömästi testialustaan, jossa suoritettiin integraatiotestaus ennen toiminnon julkaisua.

Ominaisuuksia ei tulisi lisätä ennen kuin niille on oikeasti tarvetta. Käyttämättömien ominaisuuksien tekemiseen kuluu resursseja. Ohjelman laajentaminen lisäämällä tarpeettomia ominaisuuksia monimutkaistaa järjestelmää. Koodin pitäminen yksinkertaisena mahdollistaa odottamattomiin muutoksiin valmistau-

tumisen. (Wells 2009; Brewer 2001.) Tässä projektissa suunniteltiin ennalta perustoiminnot, jotka tiedettiin etukäteen varmasti tarpeellisiksi. Tämän lisäksi toteutettiin toimintoja, joita työntekijät havaitsivat tarvitsevansa. Näin vältettiin turhien ominaisuuksien lisääminen.

Tämä projekti tehtiin vain kahden henkilön työpanoksella, joten XP:tä käytettiin vain soveltaen. Kahden hengen tiimistä ei saa kuin yhden parin ja välillä koodia piti korjata yksin, kun havaittiin jokin toiminnan pysäyttävä ongelma jo käytössä olevassa versiossa.

4 KÄYTTÖLIITTYMÄT

Graafisen käyttöliittymän tekeminen vie aikaa ja siinä tulee ottaa huomioon monia asioita, mutta oikein toteutettuna se suojaa tietokantaa virheiltiltä ja on huomattavasti miellyttävämpi kuin tietokannan ohjaaminen komentoriviltä. Yleisen mukavuuden ja virheiden vähentämisen lisäksi erillinen käyttöliittymä lisää myös tietoturvallisuutta. KMT:llä käy päivän mittaan asiakkaita, jolloin ohjelma, jolla asiakastapahtumia käsitellään, on usein koko päivän auki taustalla. Tämän seurauksena jokin huolimaton näppäinyhdistelmä tai muuten huolimaton käyttö saattaisi johtaa joidenkin tietojen menetykseen tai tietojen näkymiseen asiakkaille.

Vaihtoehtoina meillä oli tehdä erillinen ohjelma tietokannan käyttämistä varten tai tehdä käyttöliittymä HTML-lomakkeilla. Erillisen sovelluksen tekeminen voitaisiin toteuttaa esimerkiksi Javalla ja HTML:ään ei tarvittaisi kuin tekstieditori. Tarvitsimme yksinkertaisen ja helppokäyttöisen käyttöliittymän, joka on suhteellisen helposti toteutettavissa kummallakin.

Eroja tekotapojen ja lopputulosten välillä kuitenkin on. Javalla toteuttaessa käytäisimme Eclipse-kehitysympäristöä, jolla toiminnallisuuden ja tarkistusten tekeminen olisi helpompaa. Huonona puolena on se, että valmis ohjelma tarvitsisi asentaa erikseen kaikkiin koneisiin, joissa sitä tultaisiin käyttämään. HTML-versiossa on helppo toteuttaa graafinen osuus käyttämällä lomakepohjaa ja CSS-tiedostoa. Tämän lisäksi tarvitaan PHP:ta palvelinyhteyden muodostamiseen. Toiminnallisuuden toteuttaminen on haastavampaa, sillä PHP-versioiden välillä on eroja, eikä kotona toimivaksi todettu koodi välttämättä toimikaan oikein KMT:n palvelimella.

Päädyimme käyttämään HTML-lomakkeita pääasiallisena työkaluna käyttöliittymän tekemiseen. Ratkaiseva tekijä oli erillisen ohjelman joustamattomuus koneiden kanssa. Kaikissa KMT:n koneissa kuitenkin on selain, jolloin HTML-pohjaista järjestelmää voi käyttää.

Käyttöliittymän suunnitteluun yritimme hakea apua, mutta mitään suoraa ohjetta oman käyttöliittymän tekemiseen kuitenkaan ei ollut. Netistä löysimme kuitenkin useamman oppaan siitä, mitä täytyy ottaa huomioon, kun tekee ohjelmaa tietylle alustalle tai tietyllä ohjelmointikielellä. Näiden lisäksi parilla sivulla oli yleisesti asioita, joita pitää ottaa huomioon käyttöliittymää tehdessä. Päätimme lukea oppaita ja poimia niistä asioita, jotka saattaisivat auttaa käyttöliittymän tekemisessä. Kävimme myös silmämääräisesti läpi valmiiksi laadittuja standardeja, mutta ne olivat liian tiukkoja meidän käyttötarkoitukseemme, sillä ne on suunniteltu juuri tietylle ohjelmalle tai alustalle. Huonona puolena oppaissa on se, että niissä käsitellään vain vähän lomakkeita, joista käyttöliittymämme tulee muodostumaan. Hyvänä puolena on se, että niissä kerrotaan minkälaisia elementtejä yleisesti sijoitetaan minnekin ja mitä ihmiset odottavat käyttäessään ohjelmia.

4.1 Periaatteet

Hyvä käyttöliittymä antaa ihmisten, jotka osaavat käyttää hyvin tietokonetta, työskennellä ohjelman kanssa ilman, että saavat opastusta tai lukevat ohjekirjaa (Ambler 2010). KMT:n järjestelmässä kuitenkin on ominaisuuksia, joita ainoastaan projektipäällikkö tarvitsee ja ne poikkeavat muusta käyttöliittymästä, joten tulkitsimme, että ohjeet on hyvä tehdä. Ohjeet liitteenä 2.

Käyttöliittymää tehdessä tulee tehdä oma standardi, sillä käyttäjät ovat joka paikassa erilaisia ja heillä on erilaiset tarpeet ja vaatimukset. Tämä aiheutti paljon hämmennystä aluksi, kun yritimme etsiä yleistä standardia verkkopohjaisille käyttöliittymille. Useiden lähteiden tutkimisen tuloksena huomasimme niissä toistuvasti mainittavan, että luokaa oma standardi ennen kuin lähдете tekemään työtä (Smith & Moiser 1986; Thovtrup & Nielsen 1991; Ambler 2010; Schumaker 2011). Standardin tekemisen tarkoitus on se, että kaikki saman laitoksen, esimerkiksi KMT:n, ohjelmat on tehty samalla tavalla. Tällöin yhden niistä opittuaan pystyisi käyttämään kaikkia.

Kun käyttää käyttöliittymiä koskevia ohjeita täytyy muistaa, että kaikkea ei voi käyttää omassa käyttöliittymässään. Käyttöliittymiä ja käyttötarkoituksia on pal-

jon. Jokaiselle mahdollisuudelle ei ole omaa ohjetta ja joillekin on useita. Näitä yhdistelemällä ja lähimpänä omaa tarkoitusta olevia ohjeita käyttämällä saadaan tehtyä itselle standardit, joita noudattamalla käyttöliittymästä saadaan johdonmukainen ja selkeä kokonaisuus.

Hyvät ohjeet voidaan tulkita useammalla tavalla, jotta ne sopisivat useampaan käyttötarkoitukseen. Parhaatkaan ohjeet eivät kuitenkaan vastaa käytännön kokemusta. Parhaiten käyttöliittymää voidaankin arvioida käytön kautta ja käyttäjäpalautteet ovat tärkeä osa käyttöliittymän ylläpitoa. (Smith & Moiser 1986.)

Järjestelmän ulkoasua suunnitellessa pitää myös erotella informaatio datasta. Informaatio on se osa datasta, josta on hyötyä käyttäjälle. Data on kaikki järjestelmän sisältämä ja tarvitsema tieto. Mikäli aiottu informaatio on esitetty epäselvästi, se muuttuu dataksi, sillä siitä ei ole käyttäjälle hyötyä. Tämän ymmärtäminen yleisesti auttaa järjestelmän tekijöiden ja käyttäjien yhteistyötä. Jos käyttäjä ei löydä järjestelmästä kaikkea tarvitsemaansa informaatiota, järjestelmän tekijä käy katsomassa miltä sivulta kyseinen data löytyy ja ilmoittaa informaation löytyvän järjestelmästä. Ongelma todennäköisimmin on siinä, että data on esillä epäselvästi. (Smith & Moiser 1986.)

Huolellinen suunnittelu on erittäin tärkeää, sillä huonosti suunniteltu käyttöliittymä vaikeuttaa ja hidastaa työn tekoa. Käyttöliittymän perimmäinen tarkoitus on helpottaa asioiden tekemistä. Jos ohjelmaa on vaikea tai hidas käyttää, saattavat käyttäjät siirtyä tekemään samat asiat mieluummin manuaalisesti kuin hitaalla ohjelmalla. Joissain tilanteissa, kuten esimerkiksi KMT:llä, manuaalinen tekeminen olisi erittäin hankalaa, joten ohjelman kokonaan käyttämättä jättäminen ei onnistu. Tällaisissa tapauksissa huono käyttöliittymä aiheuttaa sekaannuksia ja virheitä ohjelmassa. (Smith & Moiser 1986.)

Järjestelmän pitäisi antaa käyttäjälle palautetta käyttäjän tekemistä toiminnoista, järjestelmään tai tietoihin tehdyistä muutoksista ja virheistä tai poikkeuksista, joita järjestelmään tulee tai on tullut. Kaikista muutoksista ja virheistä käyttäjän ei tarvitse tietää ja niistä kannattaa tehdä erillinen palaute tai virheilmoitus suo-

raan ylläpitäjälle. Käyttäjille tulevien ilmoitusten tulee olla selkeitä ja helposti huomattavia. (Ambler 2010.)

Datan saaminen järjestelmään pitää pystyä tekemään niin, että sama asia joudutaan kirjoittamaan vain kerran. Tähän ei lasketa mahdollisia korjauksia. Mikäli samaa tietoa joudutaan kirjoittamaan useaan kertaan, työprosessi hidastuu turhaan ja virheitä tulee enemmän. (Smith & Moiser 1986.) KMT:llä tämä hoituu uudella järjestelmällä siten, että tieto syötetään järjestelmään ja järjestelmästä tulostetaan tiedot paperilla. Vanhassa järjestelmässä tieto kirjoitettiin ensin paperille ja vasta lopuksi kirjattiin järjestelmään.

4.2 Rakenne

Käyttöliittymän rakenne pitää suunnitella siten, että se vastaa tarkoitustaan ja koostuu johdonmukaisista elementeistä, jotka ovat selkeitä ja helposti tunnistettavia käyttäjille. Samankaltaiset asiat ryhmitetään yhteen ja erilaiset asiat erotellaan selkeästi toisistaan. Kaikkien tarvittavien työkalujen tulisi olla selkeästi esillä ja häiritseviä asioita, kuten ylimääräistä tai tarpeetonta tietoa ja painikkeita, tulee välttää. Hyvissä käyttöliittymissä käyttäjää ei häiritä ylimääräisillä toimintoilla tai tiedoilla. (Ambler 2010.)

Käyttöliittymässä täytyy kuitenkin olla kaikki kentät, joita tarvitaan tiedon tallentamiseen. KMT:n tapauksessa on parempi, jos ohjelman kautta ei saa suoraan poistettua tietokannassa olevia tietoja, sillä meidän tietokannastamme ei tarvitse poistaa jo huollettuja koneita tai vaihtaa tietoja taulusta toiseen.

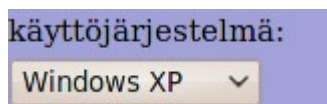
Tietojen muokkaamistoimintoa kuitenkin tarvitaan, sillä välillä tulee virheitä ja tietoa täytyy muokata. Esimerkiksi puhelinnumero ja asiakkaan nimi voivat tulla vahingossa väärin ja kaikki joudutaan kirjoittamaan uudelleen, jos kyseisiä tietoja ei pääse muokkaamaan. On käyttäjäystävällisempää olla laittamatta painikkeita, joita tarvittaisi harvoin tai ei ollenkaan. Poistopainikkeen puuttuminen vähentää vahingon mahdollisuutta huomattavasti.

Käyttöliittymän pitäisi olla joustava ja kestävä. Kestävyyttä käyttöliittymään saadaan varmistamalla, että se ei kaadu helposti. On parempi vaihtoehto, että yk-

sittäisen tapahtuman tiedot menetetään kuin se, että ohjelma on päivän poissa käytöstä. Joustavuudella tarkoitetaan sitä, että pyritään vähentämään virheiden tapahtumismahdollisuuksia. Tämä tapahtuu esimerkiksi tukemalla mahdollisimman monenlaisia syötteitä. Kestävä käyttöliittymä vähentää virheistä syntyviä kustannuksia ja mahdollistaa tiedon muokkaamisen. (Ambler 2010.)

Selkeyden kannalta on erittäin tärkeää säilyttää johdonmukaisuus kaikissa tilanteissa. Johdonmukaisuudella tarkoitetaan sitä, että esimerkiksi kaikki samanlaisen toiminnon suorittavat painikkeet ovat lähes samoissa kohdissa. Johdonmukaisuudella autetaan myös uusien ominaisuuksien kanssa, sillä jos uudet ominaisuudet toimivat samoin kuin vanhat, on käyttäjän helppo omaksua ne ilman opastusta. Johdonmukaisuuden varmistamiseksi ohjelmalle tai ohjelmistoille luodaan omat standardit. (Ambler 2010.)

Ohjelmassa kannattaa kierrättää elementtejä ja käyttäytymismalleja. Käyttäytymismallilla tarkoitetaan asiaa, joka tapahtuu aina, kun tekee tietyn asian. Kun esimerkiksi painaa tallennuspainiketta, KMT:n järjestelmässä tiedot tallentuvat tietokantaan. Tällä tavoin helpotetaan johdonmukaisuuden ylläpitämistä, mikä vähentää käyttäjien tarvetta ajatella ja muistaa asioita. Usein tehtyjen asioiden tulisi olla yksinkertaisia ja nopeita tehdä, kielen pitää olla selkeää ja käyttäjien omaa kieltä. Kaikkien ominaisuuksien pitäisi olla helposti käsillä. Täytyy myös muistaa, ettei käytä epäselviä termejä, kuten esimerkiksi ryhmä A ja ryhmä B. (Ambler 2010; Smith & Moiser 1986.) Esimerkiksi KMT:n järjestelmässä on käyttöjärjestelminä Linux ja Windows XP sen sijaan, että olisi pelkkä numero, joka pitäisi muistaa.



Kuva 3. Käyttöjärjestelmän valinta

Toisista ohjelmista ideoiden ottaminen on hyvä idea, mutta niitä kannattaa ottaa varoen. Jossakin toisessa ohjelmassa hyvin toimiva elementti saattaa vaikeut-

taa oman ohjelman käyttöä tai olla muuten sopimaton. Monesti valmiina kopioitu elementti toimii huonosti ja sitä täytyy muokata sopivaksi. (Ambler 2010.)

Elementtejä kannattaa välttää laittamasta liikaa ruudulle, sillä kaikkea ei välttämättä tarvitse tehdä yhdellä sivulla. Johdonmukaisuuden ja navigoinnin kanssa tulee ongelmia, jos elementtejä on liikaa. Myös virheet lisääntyvät, kun tulee vahingossa painettua vieressä olevaa nappia. Käyttöliittymää tehdessä kannattaa käyttää protoilun ja ketterän mallin mukaisia menetelmiä. (Ambler 2010.)

Mikäli järjestelmässä on pikanäppäimiä, ne tulee pitää selkeästi erillään tekstin kirjoittamisesta. Tämä vähentää helppojen virheiden sattumia huomattavasti, sillä varsinkin pitkiä tekstejä kirjoittaessa tulee joitain virheitä, joiden korjaamiseen tarvitaan huomattavasti näppäimiä. Hyvä tapa välttää ongelmia on laittaa toimintoja funktionäppäimiin (F1 - F12) tai käyttää tuplapainalluslogiikkaa, jolloin painetaan yhtä aikaa esimerkiksi ctrl-näppäintä ja jotain toista näppäintä, joka vastaa haluttua tarkoitusta. (Smith & Moiser 1986.)

Hyvänä esimerkkinä ctrl ja P painettuna tuovat esiin tulostuksen asetukset. KMT:n järjestelmässä ei ole käytetty erillisiä pikanäppäimiä, mutta selaimen omia pikanäppäimiä voi käyttää hyödykseen.

On tärkeää, että käytettävien elementtien välillä pääsee liikkumaan helposti. Mikäli liikkuminen ohjelman elementtien välillä on vaikeaa tai aiheuttaa ylimääräistä työtä, sen kanssa työskenteleminen muuttuu helposti turhauttavaksi. Hyvässä käyttöliittymässä on myös otettu huomioon, kenelle ohjelma tulee käyttöön. Asioita, jotka kannattaa huomioida, ovat esimerkiksi luontevuus, lukusuunta ja päivämäärien kirjoitustapa. Luontevuuden suunnittelussa auttaa ohjelman käyttäjän työtapojen tunteminen. (Ambler 2010.)

KMT:llä työskentelee it-alalla opiskelevia ihmisiä, joten meillä oli suhteellisen hyvä käsitys siitä, mitä ohjelmaa käyttävät ihmiset osaavat ja mitä he odottavat ohjelmalta. Tämän takia pystyimme tekemään lyhyitä selitteitä, jotka ovat järjestelmän käyttäjille selkeitä. Tällöin oli helpompaa miettiä elementtien sijainteja.

Käyttäjät käyttävät ohjelmaa selitteiden perusteella. Usein yhden sanan tulisi olla tarpeeksi kuvaava kertomaan mitä kenttään pitää kirjoittaa tai mitä painike tekee. Pitää myös ottaa huomioon selitteen paikka, ettei niitä voi sekoittaa keskenään. Huonot selitteet tai niiden sekava sijainti tuhoavat käyttöliittymän käytettävyyden. Kannattaa selitteiden lisäksi tarkistaa virheilmoitukset. Virheilmoitusten tulisi antaa tietoa eikä vain virhekoodia. (Ambler 2010.)

Selitteen jälkeen on hyvä olla merkki, joka kertoo, että tässä vaiheessa odotetaan käyttäjän syötettä. Merkin tulee olla sama joka välissä, koska sen vaihtelu muuttaisi lomakkeen epäselväksi. (Smith & Moiser 1986.) KMT:n järjestelmässä on erottelumerkkinä käytetty kaksoispistettä ja selitteet ovat syötekenttien yläpuolella.

Järjestelmän syötekenttien tulisi olla siinä järjestyksessä, jossa järjestelmän käyttäjä niitä ajattelee. Esimerkiksi KMT:n järjestelmässä kysytään ensin nimi ja puhelinnumero, jotka ovat yhteydessä ihmiseen, ja vasta sen jälkeen tietokoneeseen liittyvät asiat. Kun dataa syötetään järjestelmään, pitää syöttöalueen olla helposti tunnistettavissa. Täytyy myös olla selvää mitä kenttään syötetään ja missä muodossa. (Smith & Moiser 1986.)

KMT:n uudessa järjestelmässä on selkeät tekstilaatikot datan syöttämistä varten. Pidempitä tekstejä kuten vikaselostetta varten on olemassa tekstialueet, joille voi vapaasti kirjoittaa. Kuvassa 4 on esimerkki tekstilaatikosta ja tekstialueesta. Kummassakin pystyy kirjoitettuja asioita muokkaamaan ja poistamaan.



nimi:
h

vastaanottaja:
Hannu

ongelma:

Kuva 4. Tekstilaatikot

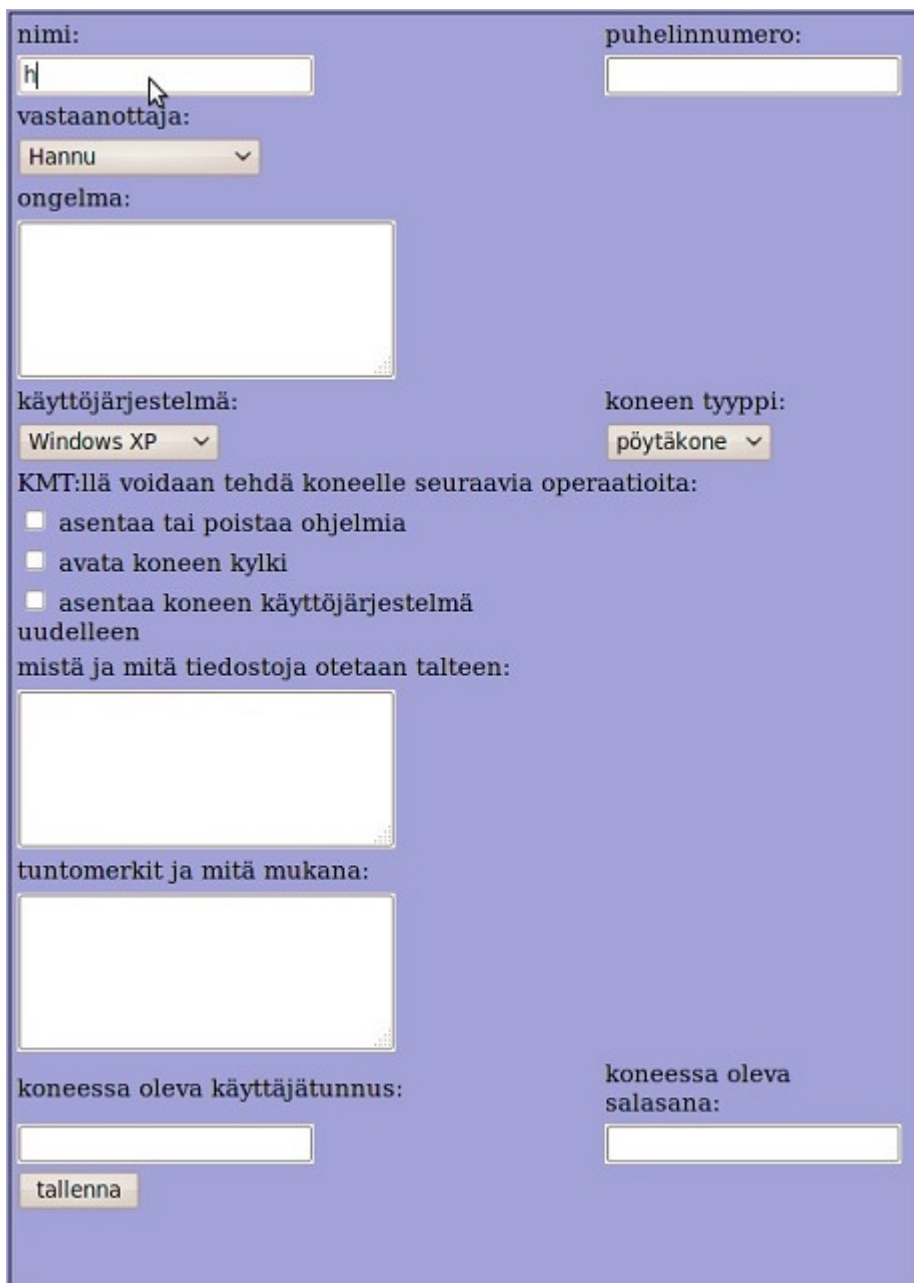
Tiedon syöttämävaiheessa tietoa täytyy pystyä muokkaamaan, jotta mahdolliset virheet voidaan korjata ennen kuin tieto pääsee tietokantaan asti. Tiedon muokkaamisen pitää onnistua kaikissa syöttökentissä samalla tavalla, jotta virheitä korjattaessa ei tulisi lisää virheitä. (Smith & Moiser 1986.) KMT:n uudessa järjestelmässä tieto syötetään erilaisiin tekstikenttiin, jolloin tiedon syöttäminen ja muokkaaminen on luonnollista ja johdonmukaista.

Pitää varmistaa, että syötteille on tarpeeksi tilaa, jotta sitä on helppo lukea ja korjata mahdolliset virheet. Tekstiä ei pitäisi joutua kelaamaan edestakaisin, koska käyttäjän ajatukset harhautuvat tekstin tarkastuksesta tekstin paikan pitämiseen kelattaessa. (Smith & Moiser 1986.) KMT:n järjestelmässä kaikki tekstikentät ovat kerralla näkyvissä ja niihin on lisätty pituusrajoitukset niin, että ne tulostuvat hyvin yhdelle sivulle.

Käyttäjän pitää antaa itse määrittää, koska hän on valmis. Tämä voidaan saavuttaa esimerkiksi tallennuspainikkeella. Tallennuspainikkeen täytyy myös olla oma painikkeensa, eli sitä ei saa yhdistää mihinkään muuhun toimintoon. Tallennuspainikkeen pitää aina olla samankaltainen ja erotettavissa muita asioita tekevästä painikkeista. (Smith & Moiser 1986.)

Kun käyttäjä täyttää lomakepohjaista tiedonsyöttöalustaa, on järkevämpää laittaa loppuun vain yksi tallennus usean sijasta. Mikäli isolla lomakkeella on paljon

eri kohtia, joita pitää erikseen tallentaa, siitä tulee sekava eikä lomake anna hyvää mahdollisuutta rauhassa läpi käydä kohtia kokonaisuutena. Lisäksi liian suuri tallennuspainikkeiden määrä saattaa aiheuttaa sen, että jotain painiketta unohtaa painaa, jolloin kyseistä tietoa ei tallennettaisi. (Smith & Moiser 1986.) KMT:n järjestelmässä joka lomakkeessa on erillinen painike, jolla tiedon saa tallennettua eikä järjestelmä tee sitä itseksensä.



nimi: puhelinnumero:

vastaanottaja:

ongelma:

käyttöjärjestelmä: koneen tyyppi:

KMT:llä voidaan tehdä koneelle seuraavia operaatioita:

- asentaa tai poistaa ohjelmia
- avata koneen kylki
- asentaa koneen käyttöjärjestelmä uudelleen

mistä ja mitä tiedostoja otetaan talteen:

tuntomerkit ja mitä mukana:

koneessa oleva käyttäjätunnus: koneessa oleva salasana:

Kuva 5. Vastaanottolomake

Kuvassa 5 on vastaanottolomake, joka on toteutettu käyttäen hyväksi havaitsemiamme ohjeita. Kaikki selitteet on tehty samalla periaatteella ja kaksoispisteen jälkeen odotetaan käyttäjän syötettä. Tekstilaatikot ovat syötteille tarpeeksi isoja ja selkeästi toisistaan erilleen sijoitettuna siten, että ne eivät mene sekaisin keskenään. Alareunassa on tallennuspainike, sillä Euroopassa lukusuunta on ylhäältä alaspäin. Koska tallennus on viimeinen asia, mitä lomakkeelle tehdään, on luonnollista, että se on viimeinen asia lomakkeella.

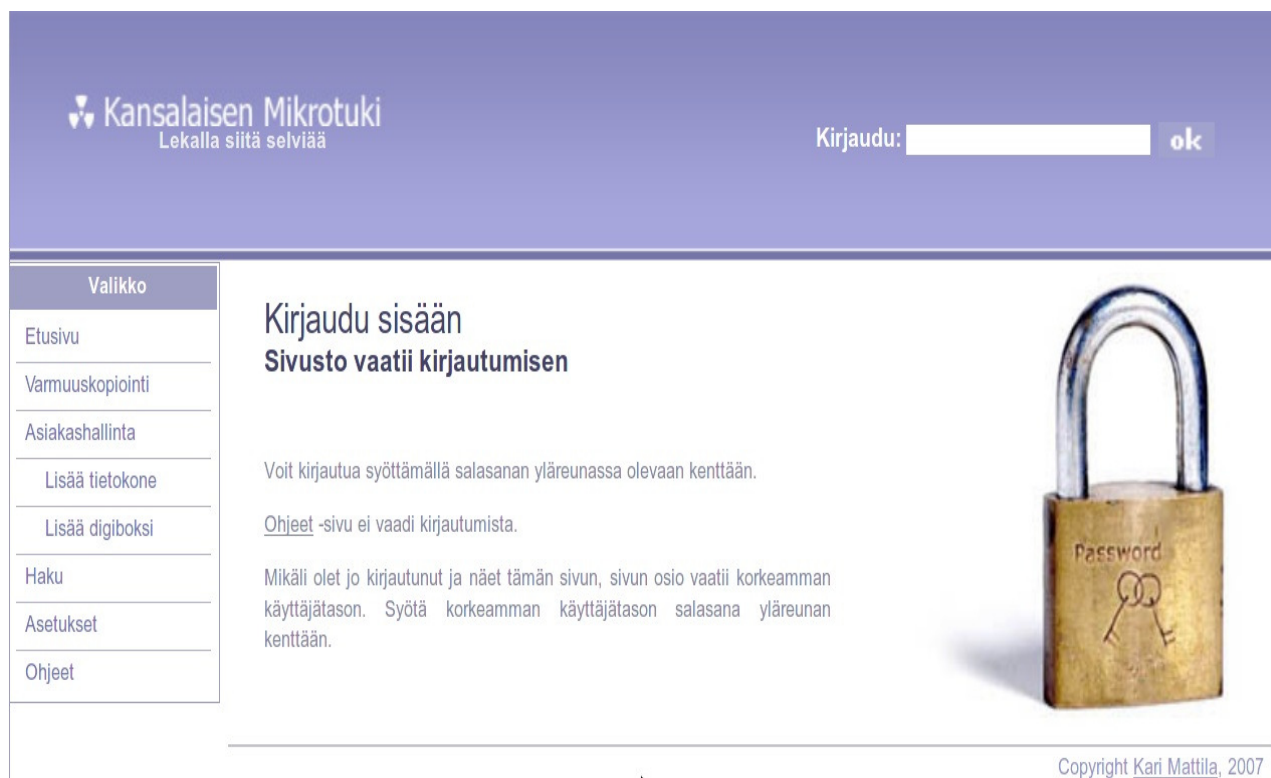
4.3 Yleisilme

Miettiessämme väritystä mietimme yleistä ilmettä ohjelmalle. Kirkkaat värit eivät olleet vaihtoehto. Niissä on vaara, että teksti jää epäselväksi ja väri käy nopeasti raskaaksi silmille. Vaihtoehtoja oli silti paljon, sillä vaaleita silmille miellyttäviä värejä riittää. Eri vaihtoehtoja läpi käydessämme löysimme värin, joka oli lähes sama kuin vanhan järjestelmän logossa ja päätimme käyttää logon väriä. Tekstin värinä käytimme perinteistä mustaa, sillä sitä on helppo lukea ja se on helppo erottaa. Musta teksti näyttää myös asialliselta verrattuna useisiin vaihtoehtoihin.



Kuva 6. Etusivu

Kuvassa 6 on järjestelmän etusivu, jossa valikko ja selitteet. Valitut sivut aukeavat keskelle ja valikko pysyy sivussa antaen hyvän mahdollisuuden siirtyä lomakkeesta toiselle.

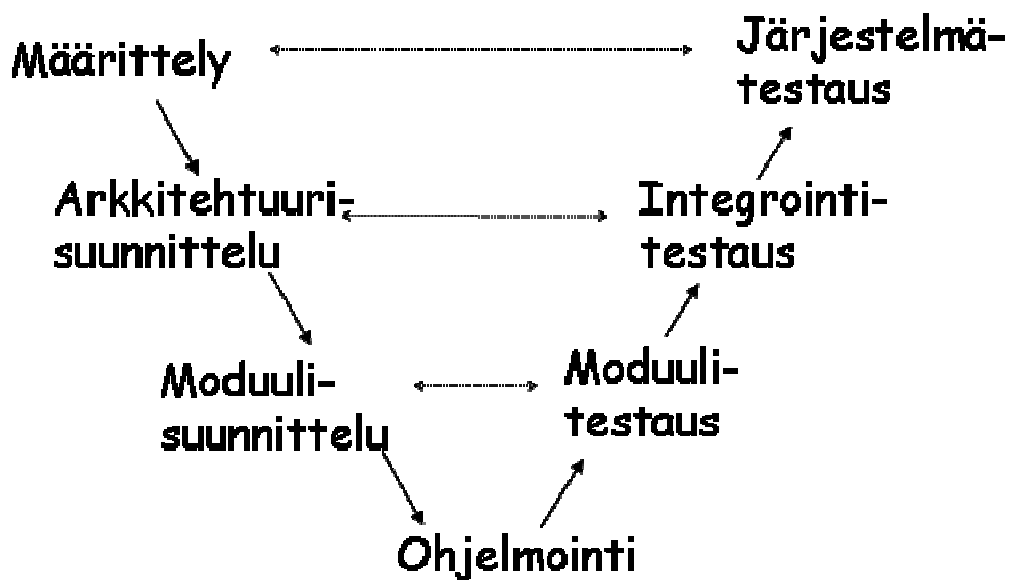


Kuva 7. Kansalaisen mikrotuen vanha järjestelmä

Kuvassa 7 on KMT:n vanhan järjestelmän etusivu, etusivu on selkeä ja helppokäyttöinen. Ylhäällä on kirjautumisruutu ja keskellä ohjeet sivun käyttöön. Uudessa järjestelmässä elementit pidettiin pääasiassa samoilla paikoilla kuin vanhassa.

5 TESTAUS

Ohjelmiston kehitysjajasta ainakin puolet on tavalla tai toisella testausta. Ohjelmoijat testaavat koodiaan monilla tavoilla ennen kuin julkistavat sen muulle tiimille. Tämän tarkoituksena on varmistaa, että muokattu koodi toimii. Kuitenkin varsinaisen ohjelmistotestauksen tarkoituksena on virheiden löytäminen ja niiden korjaaminen. Testaus on jaettu useaan vaiheeseen ja se voidaan yleisesti toteuttaa kahdella erilaisella peruslähestymistavalla. Usein testaus suoritetaan ns. V-mallin mukaisesti. (Haikala & Märijärvi 2000, 265-266, 270.) Ketteriä menetelmiä käyttäessä järjestelmälle ei kirjoiteta määrittelydokumenteja, joten järjestelmätestaus suoritetaan palavereissa sovittujen määrittelyjen perusteella.



Kuvio 1. Testauksen V-malli (Haikala & Märijärvi 2000, 271).

Moduulitestauksessa testataan yksittäinen moduuli, joka koostuu yleensä noin 100-1000 koodirivistä. Moduulitestausta varten voidaan joutua tekemään tynkämoduuleja, jotka simuloivat muiden moduulien toimintaa. Testauksen kannalta tehokkainta olisi tehdä ensin alimman tason moduulit, joita muut moduulit toiminnassaan tarvitsevat. Tämä ei valitettavasti kuitenkaan ole aina mahdollis-

ta. Testauksen suorittaa yleensä moduulin toteuttaja. (Haikala & Märijärvi 2000, 271; Zabir 2011.)

Integroititestauksessa tarkastellaan moduulien välisiä rajapintoja. Testauksen aikana yhdistellään moduuleja ja tämä yleensä tapahtuukin rinnan moduulitestauksen kanssa. Yleisimmin integrointi tehdään kokoavasti alimman tason moduuleista ylöspäin. (Haikala & Märijärvi 2000, 272; Zabir 2011.)

Järjestelmätestauksessa varmistetaan, että koko järjestelmä toimii määrittelyn mukaan. Viimeistään tässä vaiheessa testaajien tulisi olla kehityksestä irrallisia henkilöitä. Myös järjestelmän ei-toiminnalliset ominaisuudet, kuten esimerkiksi luotettavuus ja käytettävyys, testataan tässä vaiheessa. (Haikala & Märijärvi 2000, 272.)

Käytettävyystestaus saattaa kestää koko ohjelmiston kehitysajan. Usein jo määrittelyvaiheessa osa tulevista käyttäjistä valitaan testaamaan käyttöliittymän prototyyppiä valvotussa tilanteessa, jossa suoritetaan tiettyjä tehtäviä. Tämän seuraamisen perusteella käytettävyydestä vastaavat henkilöt osaavat kiinnittää huomionsa oikeisiin epäkohtiin. Tämän testauksen sijaan, tai paremmassa tapauksessa lisäksi, voidaan käyttää myös käytettävyysasiantuntijoita arvioimaan tilannetta. (Haikala & Märijärvi 2000, 273.)

Testitapausten valintaan voidaan käyttää useampaa keinoa. Ensimmäinen vaihtoehto on lasilaatikko, jolloin testitapaukset valitaan tutkimalla ohjelmiston koodia. Tällöin voidaan päätellä, mitkä tilanteet luultavimmin aiheuttavat ongelmia ja mitkä tilanteet käyttäytyvät ohjelmiston sisällä täysin samalla tavalla. Ensimmäiset testaukset suorittaa ohjelmoija itse, joten näiden kanssa lasilaatikkotestaus on yleistä. Mahdollisesti koko moduulitestaus voidaan suorittaa lasilaatikkona. (Haikala & Märijärvi 2000, 273-274; Satakar 2011.)

Toinen vaihtoehto on mustalaatikko, jossa testaus tapahtuu pelkästään määrittelydokumentin perusteella. Mustalaatikossa syöteavaruus pitää jakaa sopiviin osiin syötteiden ominaisuuksien perusteella. Esimerkiksi epäkelvot syötteet ja raja-arvojen ulkopuolella olevat syötteet ovat hyviä tilanteita. Myös juuri rajalla olevat arvot ovat tärkeitä testitapauksia. Kelvollisten syötteiden jakaminen sopi-

viin osiin on vaikeaa, sillä testaajan silmissä yhtenäiseltä näyttävät arvot saattavat toimia ohjelmiston sisällä eri tavalla. (Haikala & Märijärvi 2000, 273-274; Satalkar 2011.)

Kolmas vaihtoehto on edellisten välimuoto harmaalaatikko. Tällöin testitapausten valintaan käytetään tietoa ohjelmiston toteutusperiaatteista. Tällaisia ovat esimerkiksi muuttujien tyypit ja syötteiden lukutapa. Yleisesti testaus alkaa lasilaatikkona ja V-mallissa ylöspäin mentäessä testaus lähestyy mustalaatikkoa. (Haikala & Märijärvi 2000, 273-274.)

Ketteriä menetelmiä käyttäessä ohjelmaversioita julkaistaan asiakkaalle usein ja tämä aiheuttaa testaukselle erilaisen paineen kuin perinteinen vesiputousmalli. Tiukassa aikataulussa testauksen löytämiä virheitä ei mahdollisesti ehditä lähettämään takaisin ohjelmoijille korjattavaksi. Agile Alliance uskoo, että jos kehittäjät toimivat läheisesti asiakkaiden kanssa, erillistä testaustiimiä ei tarvita. Kuitenkin ohjelmoijien itse testaamaan koodiin jää helpommin virheitä kuin erillisen testaajan testaamaan, sillä testaaja voi tutkia ohjelmaa täysin objektiivisesti. (Malik & Nawaz 2008, 10-12.) Projektissa ohjelmoijat testasivat järjestelmän pääasiassa itse, sillä muuta henkilöstöä ei ollut käytettävissä. Välillä apuna oli KMT:n henkilöstöä, joilta sai välittömästi palautetta esimerkiksi käyttöliittymästä.

Extreme Programmingissa kaikki moduulit ajetaan automaattisten testien läpi. Nämä testit tehdään ennen toiminnallisuuden ohjelmointia. (Wells 2009.) Tämän takia Extreme Programmingin testaus on mustalaatikkotestausta, jossa ei ole määrittelydokumenttia, vaan määrittely on sovittu jakson alkupalaverissa.

Normaalista Extreme Programmingista poiketen tämän projektin moduulitestausta ja integointitestausta suoritettiin lasilaatikkona. Järjestelmä koostuu lomakkeista, joiden toiminnallisuus on ohjelmoitu PHP:llä, eikä tällaisten lomakkeiden testaukseen saa tehtyä automaattisia työkaluja.

Lomakkeen testauksen alussa täytettiin lomake laittamalla jokaiseen kenttään odotusarvon mukainen syöte. Tämän jälkeen tehtiin kolme testiajoa, joissa laitettiin jokaiseen kenttään ensin pelkkiä numeroita, sitten pelkkiä kirjaimia ja lopuksi näiden lisäksi erikoismerkkejä sisältäviä syötteitä ja SQL-injektion yritys.

Tämän jälkeen ajettiin vielä sattumanvaraisia syötteitä, joita luotiin tarpeen mukaan testaajan haluamalla tavalla. Näin varmistettiin, ettei yksi testiajo mennyt vain vahingossa läpi ja tämän vuoksi virheellisesti toimiva lomake pääse käyttöön.

Lomakkeen toimimattomuus oli suurimmassa osassa lomakkeita ensimmäisenä ongelmana. Kun lomake saatiin kirjoitusvirheiden korjaamisen – yleisimmin lainausmerkkien puute – jälkeen toimimaan odotusten mukaisilla arvoilla, millään syötteillä ei tullut testauksen aikana odottamattomia virheitä. Yksi muokkaus aiheutti kuitenkin ongelman, koska sen testaaminen jäi jostain syystä kokonaan tekemättä. Osa erikoismerkeistä aiheutti tämän jälkeen virhetilanteen ja tämä ilmeni vasta käytön aikana.

Suurin osa normaalin toiminnan ongelmista tuli päivämäärien kanssa. Kirjoitusvirheiden takia järjestelmä saattoi vähentää nykyisestä vuodesta kuukauden ja päivän, minkä vuoksi asioita tallentui pahimmillaan vuoteen 1986.

Toiminnallisuuden testauksen lisäksi skandinaavisten merkkien näkyminen lomakkeilla tuotti ongelmia. Ongelmaan oli ratkaisuna merkistökodeauksen vaihtaminen kaikissa järjestelmän osissa – www-palvelin, SQL-palvelin ja koodin sisältävät tiedostot – samaksi.

6 TIETOTURVA

Tietoturva koostuu kolmesta perusasiasta, jotka ovat tiedon eheys, saatavuus ja luottamuksellisuus. Tiedon eheys tarkoittaa sitä, että tieto on oikeaa ja luettavassa muodossa. Se ei saa myöskään muuttua tarkoituksesta. Saatavuudella tarkoitetaan, että tietoon pitää päästä käsiksi. Luottamuksellisuudella tarkoitetaan, että tieto on turvassa tahoilta joille se ei kuulu. (Watson Business Systems Ltd 2007.)

Tietoturvan ongelmana on juuri kolmen perusasian tasapainottaminen sopivaksi. Mikäli luottamuksellisuuteen ollaan panostettu liikaa, on tieto tavoittamattomissa myös niille jotka sitä tarvitsevat tai niin vaikea päästä käsiksi, että se haittaa päivittäistä toimintaa. Toisaalta jos tietoihin pääsee liian helposti käsiksi, sitä etsivät asiaan kuulumattomat henkilöt saisivat tiedon käsiinsä. Tiedon eheyteen vaikuttavat erilaiset lokitiedot, jotka kertovat ihmisten toimista kyseisten tietojen kanssa ja niiden avulla voidaan jäljittää tehdyt muutokset.

Tietoturvan perusteita voidaan täydentää kiistämättömyydellä, tunnistettavuudella ja todennuksella. Kiistämättömyydellä tarkoitetaan, että henkilön vaikuttaessa tietoihin siitä jää jälki, jolla voidaan todeta, että kyseinen henkilö on tehnyt kyseisen muutoksen. Tunnistuksella tarkoitetaan sitä, että kaikki henkilöt, jotka pääsevät tietoihin käsiksi, ovat liitettävissä tiettyyn käyttäjätunnukseen. Todennuksella tarkoitetaan sitä, että kaikki osapuolet voidaan varmentaa tietyiksi henkilöiksi.

Tietoturvan perusteista on myös malleja, joissa on käytetty viittä kohtaa. Näissä todennus ja tunnistettavuus on yhdistetty yhdeksi kohdaksi, jolloin sen nimeksi tulee auditointi. (Watson Business Systems Ltd 2007.)

6.1 Tietoturvapoliitikat

Yrityksillä on tarkka tietoturvapoliitikka, joka auttaa työntekijöitä ymmärtämään ja toteuttamaan omaa osaansa tietoturvan tapahtumista. Esimerkkinä tietoturvapoliitikan kohdasta voitaisiin käyttää sääntöä "älä kytke omaa muistitikkuasi

työkoneisiin". Tällä ohjeella pyritään välttämään ihmisten kotona mahdollisesti saastuneilta koneilta virusten tai muiden haittaohjelmien, jotka heikentäisivät tietoturvallisuutta yrityksessä ja mahdollistaisivat esimerkiksi tietomurtoja, leviäminen työympäristöön. Pitää myös muistaa, etteivät kaikki tietoturvan säännöt liity mitenkään tietokoneisiin. Esimerkkinä "silppua vanhat kokouspöytäkirjat" tai "älä kerro kenellekään työsi yksityiskohdista". Nämä ohjeet ovat vanhoja ja sellaisia, joita usein pidetään itsestäänselvyyksinä. Ne myös osoittavat, että niin kauan kuin yrityksillä on salattavaa tietoa, niillä on tietoturvariskejä.

KMT:llä tietoturvan ohjeita ovat esimerkiksi seuraavat:

- älä katso asiakkaan tiedostoja tai kuvia
- älä päästä asiakkaita työtilaan
- älä päästä asiakasta KMT:n koneelle valvomatta
- älä jätä työtilaa yksin työpäivän aikana
- muista lukita ovet ja sammuttaa koneet tilan jäädessä tyhjäksi.

Tietoturvapolitiikoista ei kuitenkaan olisi hyötyä, mikäli ihmiset eivät tietäisi niistä. Yritysten tulisi pitää säännöllisin väliajoin luentoja tai tilaisuuksia, joissa ihmisiä muistutettaisiin näistä politiikoista ja kerrottaisiin uusista. Mikäli halutaan, että ihmiset pysyvät valppaana tietoturvan suhteen, täytyy heitä välillä muistuttaa. Jos työntekijälle ei ole kukaan kertonut, että jotain asiaa ei saa tehdä, sitä ei tulla noudattamaan ja tietoturvapolitiikkaan käytetyt resurssit menevät hukkaan.

KMT:llä tietoturva-asioista ja säännöistä kerrotaan, kun ihminen tulee ensimmäisen kerran töihin tai kurssille, ja sen jälkeen nämä asiat pitäisi muistaa. Uutta henkilöstöä tulee kuitenkin usein ja perehdytyksien aikana vanhatkin työntekijät kuulevat asiat uudelleen. KMT:n käytännöt ovat yksinkertaisia ja selkeitä, joten niiden muistaminen ei ole vaikeaa, mutta kurssia voidaan suorittaa vuosien mittaan. Tällöin olisi hyvä, jos käytännöt olisivat esimerkiksi jossain näkyvillä.

Tietoturvasta kannattaa myös muistaa, ettei se ole pelkästään tietokoneiden sisällä tapahtuvaa tai muistitikuilla kulkevaa, vaan myös huolellisuus ja tavat vaikuttavat siihen. Esimerkiksi ovien lukkoon laittaminen vaikuttaa myönteisesti tietoturvaan, sillä jos joku voi vain kävellä sisään yritykseen ja ottaa tarvitsemansa, ei palomureista ole hyötyä. Tapoina voidaan mainita esimerkiksi usein tallentaminen, joka auttaa pitkien artikkelien kanssa, sillä sähkökatkoja tai jotain muuta yllättävää saattaa tapahtua. Jos menee sammuttamaan ohjelman tallentamatta, voivat yhden päivän työt mennä kokonaan hukkaan.

6.2 Tietoturva KMT:llä

KMT:n uuden järjestelmän kanssa tietoturva-asioita piti miettiä hieman eri näkökulmasta kuin normaalisti. Koska järjestelmä tuli pelkästään sisäiseen käyttöön, piti miettiä salasanojen käyttöä ja niiden vaikutusta. Mikäli laitettaisiin salasana, sen pitäisi olla helposti muistettava, jottei se hidastaisi työntekoa ja ettei sitä tarvitsisi kysellä jatkuvasti. Salasana saattaisi olla hetken kuluttua myös jollain lapulla, jotta sen voisi tarkistaa mikäli se sattuisi unohtumaan, mikä jo itsessään on tietoturvariski. Toinen vaihtoehto on luoda jokaiselle työntekijälle järjestelmään käyttäjätunnus, mutta osa työntekijöistä saattaa tulla takaisin parinkin vuoden tauon jälkeen ja silloin omaa salasanaa ei enää muista. KMT:n uudessa järjestelmässä ei ole edellä mainituista syistä salasanvoja.

KMT:n tärkeimmät tietoturva-asiat koskevat ihmisten tietoja ja koneita. KMT:llä pystytään pitämään tiedot hyvin salassa, kunhan asiakkaita ei päästetä vastaanottopuolelta työntekijöiden puolelle tai käyttämään valvomatta KMT:n verkkoon kytkettyä konetta. Mikäli asiakkaat pääsisivät vapaasti työntekijöiden puolelle, he voisivat nähdä ihmisten tiedostoja. Mikäli he pääsevät koneelle, joka on KMT:n verkossa, heillä on mahdollisuus päästä järjestelmään tutkimaan kaikkia tietoja, mikäli tietävät palvelimen osoitteen. Palvelimen tietojen ei pitäisi olla ulkopuolisten tiedossa, mutta ihminen on tietoturvan heikoin lenkki.

6.3 Etäseuranta

KMT:n uuteen järjestelmään alunperin suunnittelimme lisäävämmä etäseurantamahdollisuuden, jonka avulla asiakkaat pääsisivät katsomaan huollossa olevan koneen tilaa. Tällaista ominaisuutta oltaisiin tarvittu, sillä asiakkaat soittavat jatkuvasti KMT:lle huollossa olevien koneiden perässä. Olisi hyvä, jos olisi vaihtoehtoinen tapa selvittää koneen tila, sillä vaikka kaikilla asiakkailla ei olekaan toista konetta, vähentäisi se silti soittojen määrää. Tätä mahdollisuutta ei kuitenkaan lisätty, sillä KMT:n verkkoon ei pääse ulkoverkosta mitenkään.

Meitä kuitenkin kiinnosti, millaisia tietoturva-asioita olisi pitänyt ottaa huomioon tällaista ominaisuutta tehdessä. Mietimme yhdessä millainen ominaisuudesta olisi tullut, millaista salasanelitiikkaa käyttäisimme ja millaisia tietoturva-asioita niihin liittyy.

Seurannan toteutuksen osalta meillä oli kaksi vaihtoehtoa: lista huollossa tällä hetkellä olevista koneista tai paneeli, johon voi kirjautua sisään omalla väliaikaisella tunnuksella. Listauksessa näkyisi vain koneen huoltonumero, joka kerrottaisiin asiakkaalle etukäteen ja koneen tila joka olisi jonossa, huollossa tai valmis. Kirjauttavassa järjestelmässä asiakkaalle annettaisiin lappu, jossa lukee hänen huoltonumeronsa ja väliaikainen salasana, jolla hän pääsee katsomaan koneensa tilaa ja kommentteja, joita huoltaja on antanut, esimerkiksi että käyttöjärjestelmä on asennettu uudelleen.

Listausvaihtoehto on tietoturvallisempi, sillä siinä ei ole mahdollista saada väärää tietoa mitenkään. Myöskään ei anneta käyttäjän syöttää mitään mihinkään kenttään eikä erillistä salasanan generointi- ja poisto-ominaisuutta tarvittaisi. Toisaalta kaikille näkyvään listaukseen ei voitaisi laittaa yhtä paljon tietoa kuin erikseen kirjaututtavaan versioon, missä asiakas näkisi vain oman koneensa asiat. Vaihtoehtoja voisi myös yhdistää, jolloin jos asiakas haluaisi nähdä vain sen, onko kone jo työn alla, ei tarvitsisi erikseen kirjautua. Jos asiakas haluaisi tietää koneensa tilasta enemmän, hänen tarvitsisi vain painaa linkkiä ja kirjoittaa hänelle annettu salasana.

Mikäli olisimme tehneet etäseurantamahdollisuuden, olisimme tehneet erilaisia ratkaisuja. Esimerkiksi emme olisi käyttäneet PHP:n mysql-laajennusta, vaan mysql:itä. Tämä johtuu siitä, että mysql on vanhentunut ja sen käyttäminen sisältää tietoturvariskejä (Sani 2011a).

Olisimme myös tehneet tarkemmat rajoitukset järjestelmän käyttöön. Ulkoverkosta olisi päässyt vain katsomaan tietoja, muokkauslomakkeet olisi pidetty vain sisäverkossa. Suojauduimme joka tapauksessa SQL-injektioilta, jotka ovat vaarallisimpia hyökkäyksiä tietokantapohjaisia järjestelmiä vastaan. SQL-injektiota vastaan suojauduimme suodattamalla tarkasti kaikki syötteet. Käytimme suojausta, etteivät työntekijät pysty vahingossa syöttämään haitallisia komentoja tai merkkejä järjestelmään. (Sani 2011b.)

Löysimme listan 25 vaarallisimmasta tietoturvirheestä, joita ohjelmoitessa tulee tehtyä (Christey 2011). Web-lomakkeita käyttäessä suurin osa virheistä on otettu huomioon palvelinohjelmistossa, jolloin emme olisi edes voineet tehdä niitä. Tunnistautumiseen ja varmentamiseen liittyvät ongelmat olisimme saaneet hoidettua sillä, ettei mitään tietoja pääse ulkoverkosta muokkaamaan, vaan kirjautumalla pääsee näkemään vain oman koneensa tietoja.

7 JOHTOPÄÄTÖKSET

Tämä projekti näytti molemmille tekijöilleen monien ryhmätyön perusteiden tärkeyden. Uskoimme työn etenevän omalla painollaan työnjaon jälkeen, mutta selkeän aikataulun puute venytti etenemistä. Työn empiirisellä osalla oli aikataulu, jonka aikana järjestelmä otettiin käyttöön.

Projektin loppupuolella kävi ilmi, ettemme olleet ottaneet huomioon KMT:n suomea osaamattomia työntekijöitä tai asiakkaita. Järjestelmässä on monikielisyiden kanssa suuri suunnitteluvirhe. Monikielisyys ei ollut alun perin vaatimuksena, joten osa tekstistä on tallennettu tietokantaan eikä sitä saa suoraan php-koodilla käännettyä toiselle kielelle. Tämän korjaaminen muuttaisi tietokannan sisältöä ja saattaisi tämän takia sotkea osan järjestelmässä jo olevasta tiedosta, joten tähän ei haluttu ryhtyä. Monikielisyys on kuitenkin KMT:ssä tärkeä ominaisuus, joten tämä lisättiin niin hyvin kuin tietokantaan koskematta pystyttiin. Tulevissa KMT:n projekteissa kannattaa määritellä yhdeksi vaatimukseksi monikielisyys.

Tämä projekti soveltui ketterän kehityksen menetelmien testaamiseen hyvin. KMT:n projektipäällikkö vaihtui työtä tehdessä kahdesti ja jokaisella oli hieman erilainen näkemys tarvittavista ominaisuuksista. Extreme Programmingin ohjeiden mukaisesti koodia yritettiin pitää yksinkertaisena, jotta uudistuvat vaatimukset saataisiin sisällytettyä järjestelmään ilman suurempia ongelmia. Kielenvaihtoa lukuun ottamatta tässä onnistuttiin hyvin.

Projekti toimi myös hyvänä ikkunana käyttöliittymäsuunnittelun maailmaan ja siihen, mitä käyttöliittymäsuunnittelijoiden pitäisi ajatella ja tehdä projektin aikana. Yllätyksenä tuli, että asioille, joita nykyisin ohjelmia käyttäessään pitää itseltänselvänä, luotiin pohja jo 25 vuotta sitten (Smith & Moiser 1986). Tietysti asiat ja ohjelmat ovat kehittyneet sen jälkeen, mutta perusasiat ovat pysyneet samanlaisina ja silloin tehdyt ohjeet ovat ilmeisesti olleet hyvässä käytössä.

Kokonaisuudessaan projektista jäi mieleen koulussa opitusta poikkeava tapa ohjelmistoprojektin ja käyttöliittymien toteuttamiseen. Ohjelmistotuotannon ja

käyttöliittymien suunnittelun kursseilla annettiin tarkat puitteet toteutuksille. Käyttöliittymää suunnitellessa sisäisten standardien luonti muuttaa puitteita. Ketterien menetelmien kanssa sekä asiakkaalla että kehittäjällä on enemmän vapauksia toteutuksessa kuin perinteisessä vesiputousmallissa.

LÄHTEET

Agile Alliance. 2001. Manifesto for Agile Software Development. Viitattu 21.2.2011
<http://www.agilemanifesto.org/>

Ambler, S. 2010. User Interface Design Tips, Techniques, and Principles. Viitattu 25.2.2011
<http://www.amblysoft.com/essays/userInterfaceDesign.html>

Brewer, J. 2001. Extreme Programming FAQ. Viitattu 3.8.2011
<http://www.jera.com/techinfo/xpfaq.html>

Christey, S. 2011. 2011 CWE/SANS Top 25 Most Dangerous Software Errors. Viitattu 1.8.2011
<http://cwe.mitre.org/top25/?2011#Listing>

Cockburn, A. 2007. Agile Software Development, Second edition. Fourth printing. Boston: Pearson Education, Inc.

Haikala, I & Märijärvi, J. 2000. Ohjelmistotuotanto. 7., uudistettu painos. Helsinki: Satku.

Kalermo, J. & Rissanen, J. 2002. Agile software development if theory and practice. Pro Gradu: tietojenkäsittelytieteiden laitos. Jyväskylä: Jyväskylän yliopisto. Saatavilla myös
<http://urn.fi/URN:NBN:fi:juu-2002888496>

Malik, K & Nawaz, A. 2008. Software testing process in agile development. Master Thesis: Department of Computer Science. Ronneby: Blekinge Institute of Technology. Saatavilla myös
[http://www.bth.se/fou/cuppsats.nsf/all/249945527e869a47c125746c0002f4e1/\\$file/Software_Testing_Process_in_Agile_Development.pdf](http://www.bth.se/fou/cuppsats.nsf/all/249945527e869a47c125746c0002f4e1/$file/Software_Testing_Process_in_Agile_Development.pdf)

Sani, I. 2011a. PHP:n tekijät: Älkää käyttäkö mysql-laajennustamme! Tietoviikko. Viitattu 1.8.2011
<http://www.tietoviikko.fi/kehittaja/phpn+tekijat+alkaa+kayttako+mysql+laajennustamme/a656999>

Sani, I. 2011b. Pahimmat koodausmokat listattiin - oletko sortunut näihin? Tietoviikko. Viitattu 1.8.2011
<http://www.tietoviikko.fi/kehittaja/pahimmat+koodausmokat+listattiin++oletko+sortunut+naihin/a652096>

Satalkar, B. 2011. White box testing vs black box testing. Viitattu 4.8.2011
<http://www.buzzle.com/articles/white-box-testing-vs-black-box-testing.html>

Schumaker, D. 2011. User Interface Standards. Viitattu 1.6.2011 <http://msdn.microsoft.com/en-us/library/aa217660%28v=office.11%29.aspx>

Smith, S. & Moiser, J. 1986. Guidelines for designing user interface software. Viitattu 20.2.2011
<http://hcibib.org/sam/>

Tehokkain työryhmä: valtaosa naisia, puolet narsisteja. Tiede 11/2010, 7.

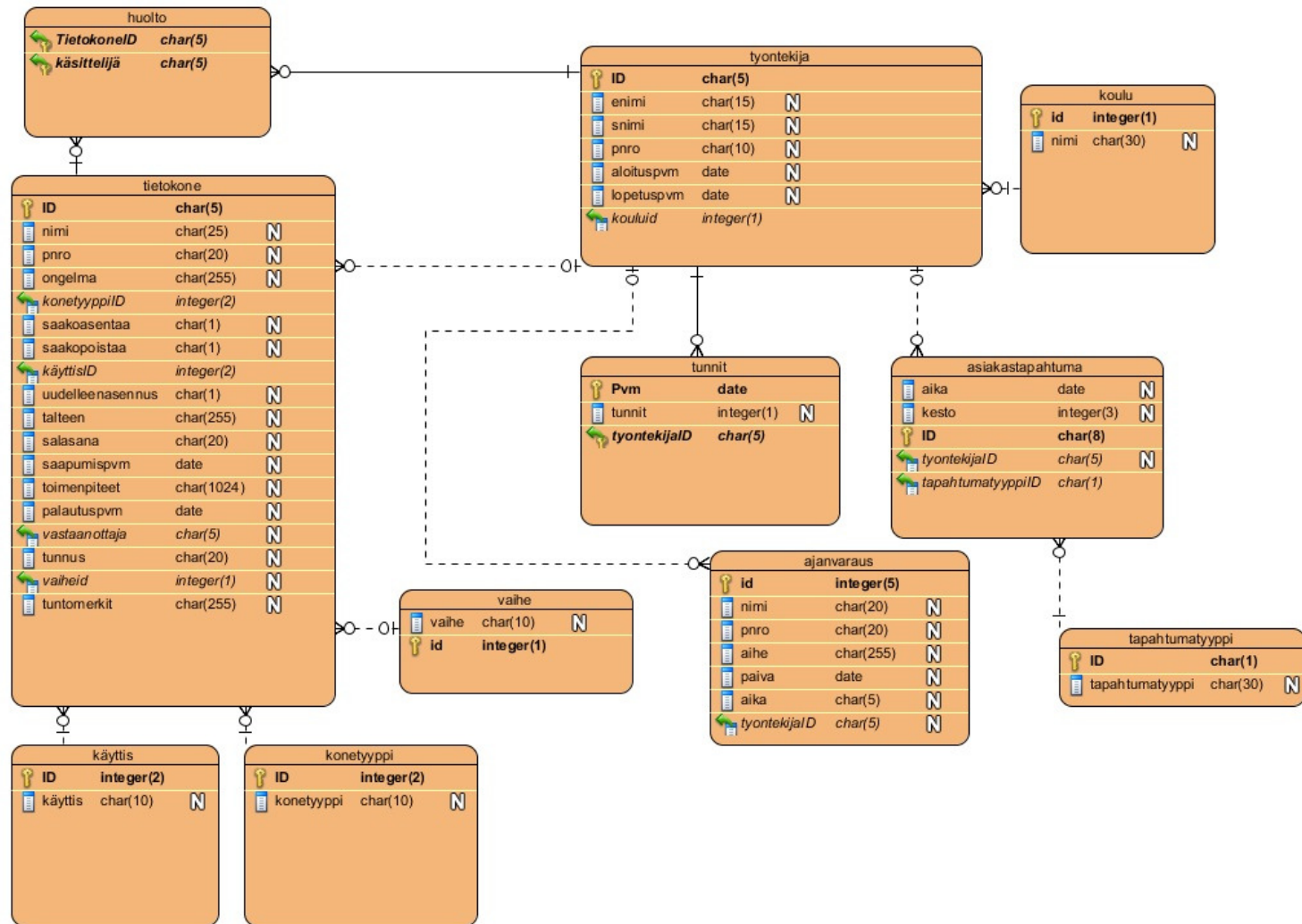
Thovtrup, H. & Nielsen, J. 1991. Assessing the Usability of a User Interface Standard. Viitattu 1.6.2011 <http://www.useit.com/papers/standards.html>

Watson Business Systems Ltd 2007. ISO 27001: An Introduction To Information, Network and Internet Security. Viitattu 2.6.2011 http://security.practitioner.com/introduction/infosec_2.htm

Wells, D. 2009. The Rules of Extreme Programming. Viitattu 19.5.2011
<http://www.extremeprogramming.org/rules.html>

Zabir, O. 2011. Unit Testing and Integration Testing in Business Applications. Viitattu 4.8.2011
<http://www.codeproject.com/KB/testing/realtesting.aspx>

Tietokannan rakenne



Järjestelmän käyttöopas

Tämän käyttöoppaan lukeminen on erittäin tärkeää jokaiselle uudelle projektipäällikölle, jotta tämä osaisi opastaa muita käytössä ja tietäisi varmasti kaikki projektipäälliköille tarkoitetut osiot. Tässä oppaassa käydään lomake lomakkeelta läpi, mitä mikäkin tekee ja lopussa käsitellään ainoastaan projektipäällikölle suunnitellut ominaisuudet.



Kuvassa järjestelmän etusivu, joka on KMT:n pääkoneella Firefoxin etusivuna. Muilta koneilta siihen pääsee osoitteessa <http://192.168.1.200/>. Sivun kohtuullisen yksiselitteinen, vasemmalla olevia linkkejä klikattaessa selitteessä lukeva lomake aukeaa oikealle. Järjestelmän kieli on mahdollista vaihtaa englanniksi (tai takaisin suomeksi) vain etusivun painikkeella.

Kansalaisen Mikrotuki

[Etusivu](#)
[Vastaanotto](#)

[Muu asiakastapahtuma](#)

[Koneen muokaus](#)

[Työntekijöiden tarkastelu](#)

[Uusi työntekijä](#)
[Ajanvaraus](#)

nimi:

puhelinnumero:

vastaanottaja:

ongelma:

käyttöjärjestelmä:

koneen tyyppi:

KMT:llä voidaan tehdä koneelle seuraavia operaatioita:

asentaa tai poistaa ohjelmia

avata koneen kylki

asentaa koneen käyttöjärjestelmä uudelleen

mistä ja mitä tiedostoja otetaan talteen:

tuntomerkit ja mitä mukana:

koneessa oleva käyttäjätunnus:

koneessa oleva salasana:

tallenna

Kuvassa vastaanottolomake.

- Asiakkaan nimi ja puhelinnumero tallennetaan vastaaviin kohtiin.
- Vastaanottaja-kohtaan valitaan se henkilö, joka ottaa koneen vastaan. Tämä henkilö valitaan aktiivisista työntekijöistä.

- Ongelma-kohtaan kirjoitetaan tarkka ongelma kuvaus. Vastaanottaja ja huoltaja saattavat olla täysin eri henkilö, joten tämän kanssa on syytä olla tarkkana!
- Käyttöjärjestelmä ja koneen tyyppi valitaan asiakkaan konetta vastaavaksi, mikäli asiakas tietää koneensa käyttöjärjestelmän. Tämä voidaan muuttaa jälkikäteen, kun huoltaja on käynnistänyt koneen ja tarkistanut asian.
- Operaatioista valitaan ne, jotka asiakas hyväksyy koneelleen tehtävän.
- Talteen otettavat tiedostot ja näiden sijainti on syytä kirjata vielä tarkemmin kuin ongelma kuvaus. Asiakkaan tärkeiden tiedostojen talteen ottaminen on mukavampi tehdä ennen formatointia kuin sen jälkeen. Jos asiakkaan mielestä koneella ei ole mitään tärkeää, kannattaa asiaa kysyä pari kertaa uudelleen, sillä aina niistä jotain löytyy kun asiakas miettii hetken.
- Koneen tuntomerkit ja mukana olevat tavarat merkitään niin täydellisesti kuin mahdollista. Kolmen mustan läppärin erottaminen toisistaan ilman muuta tietoa on mahdotonta. Jos tämän lisäksi on vielä ylimääräinen laukku, jota ei ole merkitty mihinkään, katastrofin ainekset ovat koolla.
- Lopuksi kannattaa varmistaa, että käyttäjätunnus ja salasana täsmäävät, etkä kirjoittanut niitä väärin.

Kansalaisen Mikrotuki

[Etusivu](#)
[Vastaanotto](#)

[Muu asiakastapahtuma](#)

[Koneen muokkaus](#)

[Työntekijöiden tarkastelu](#)

[Uusi työntekijä](#)
[Ajanvaraus](#)

tekijä: ▼

kesto minuutteina:

tyyppi: vastaanotto ▼

tallenna

listaa asiakastapahtumat aikaväliltä

-

listaa asiakastapahtumat

Kuvassa muu asiakastapahtuma -lomake.

- Tekijä valitaan aktiivisista työntekijöistä.
- Asiakastapahtumia tehdessä kannattaa katsoa kellosta aikaa usein ja miettiä kesto, jotta sen osaa tallentaa tietokantaan oikein.
- Asiakastapahtuman tyyppi valitaan projektipäälliköiden tallentamista vaihtoehtoista. Vastaanotto ja luovutus tallentuvat automaattisesti koneita vastaanotettaessa ja niiden valmiiksi merkitessä.
- Lomakkeella on myös mahdollista etsiä vanhoja asiakastapahtumia kirjoittamalla päivämäärät muotoon dd.mm.yyyy ja painamalla listaa asiakastapahtumat -painiketta.

Kansalaisen Mikrotuki

[Etusivu](#)
[Vastaanotto](#)

[Muu asiakastapahtuma](#)

[Koneen muokkaus](#)

[Työntekijöiden tarkastelu](#)

[Uusi työntekijä](#)
[Ajanvaraus](#)

Palauttamattomat koneet

| nimi | puhelinnumero | ongelma |
|-------------------|---------------|-------------------------------|
| K | 04. | Vikasietotilassa toimii, mutt |
| N | 04. | Laitetaan kone käyttökuntoo |
| S | 04. | Windows Vista pitäisi asenta |
| H | 04. | Mokkula ei päästä internet |
| R | 04. | Kone ei lähde käyntiin. Pit |
| P | 04. | Ei pääse windowsiin asti. |
| S | 04. | Kone ei yhtäkkiä löydä en |
| O | 04. | wlan ei ota yhteyttä reititt |
| Z | 04. | XP reinstall, updates |
| V | 04. | Kone ei käynnisty kunnolla, |
| P | 04. | Norton poistetaan, asenn |
| M | 04. | "Windows ei voinut muodostaa |
| S | 04. | Kone heittää valitusteksti |
| M | 04 | F-Secure pois, Norton tilalle |
| P | 04. | Kone ei käynnisty. Kovalevy |
| K | 04. | HP: liikaa ohjelmia, kone tö |
| M | 04 | Windows ei käynnisty. K |
| K | 04. | Avast lisenssikoodin uusimin |
| N | 04. | Verkkokortti ei toimi |
| R | 04. | Viruspäivityksen uusimine |
| H | 04. | Tietokone ei käynnisty, pyö |
| A | 04. | Näytönohjain rikki (?). "No |
| E | 04 | Ei tarkkaa diagnoosia. "Kone |
| k | 04. | Virustorjuna mennyt vanhaksi, |
| G | 04. | Käynnistysongelmia |

Hae kaikista koneista

Hakuehtona asiakkaan nimi tai puhelinnumero

Kuvassa koneenhakulomake.

- Oletuksena hakutuloksena ovat palauttamattomat koneet.
- Listauksessa on asiakkaan nimi, puhelinnumero ja lyhennetty kuvaus koneen ongelmasta.
- Nimeä klikkaamalla pääsee muokkaamaan kyseistä konetta.
- Kaikista koneista voi hakea alareunan hakupainikkeella.

[Työntekijöiden tarkastelu](#)

[Uusi työntekijä Ajanvaraus](#)

käyttöjärjestelmä
Windows XP ▾

koneen tyyppi:
pöytäkone ▾

KMT:llä voidaan tehdä koneelle seuraavia operaatioita:

asentaa tai poistaa ohjelmia

avata koneen kylki

asentaa koneen käyttöjärjestelmä uudelleen

mistä ja mitä tiedostoja otetaan talteen:

tuntomerkit ja mitä mukana:

koneessa oleva käyttäjätunnus:

koneessa oleva salasana:

työvaihe:
jonossa ▾

käsittelijät
 ▾

lisää käsittelijä

koneelle tehdyt toimenpiteet:

saapumispäivä

palautuspäivä


tallenna

[tulosta asiakastietolomake](#)

Kuvassa koneenmuokkauslomake.

- Muokkauslomakkeen yläosa on samanlainen kuin vastaanottolomakkeen
- Työvaihe päivitetään aina, kun kone otetaan jonosta työn alle tai sen status muuttuu.

- Käsittelijät valitaan aktiivisista työntekijöistä ja tallennetaan napilla. Nämä on tarkoitettu auttamaan projektipäällikköä selvittämään, keneltä voi tiedustella tapahtumia reklamaatiotilanteissa.
- Tehdyt toimenpiteet tulisi listata kattavasti, mutta samaan aikaan lyhyesti ja ymmärrettävästi. Tämä tieto menee asiakkaalle, joten käytettyä kieltä kannattaa miettiä tarkkaan.
- Saapumispäivä tulee automaattisesti konetta vastaanotettaessa, palautuspäivä taas laitettaessa kone luovutetuksi.
- Kun kone on valmis, tulosta sille asiakastietolomake, joka annetaan asiakkaalle tämän noutaessa konetta.



Kansalaisen Mikrotuki

[Etusivu](#)
[Vastaanotto](#)

[Muu asiakastapahtuma](#)

[Koneen muokkaus](#)

[Työntekijöiden tarkastelu](#)

[Uusi työntekijä](#)
[Ajanvaraus](#)

työntekijä:

Hae

Kuvassa työntekijöiden valintalomake. Työntekijä-kohdasta valitaan KMT:n työntekijä. Haku näyttää työntekijästä tallennetut tiedot.

Kansalaisen Mikrotuki

| | | | | |
|--|---|--|---|-----------------------------------|
| Etusivu Vastaanotto Muu asiakastapahtuma Koneen muokaus Työntekijöiden tarkastelu Uusi työntekijä Ajanvaraus | etunimi: | sukunimi: | id: | |
| | <input type="text"/> | <input type="text"/> | 34 | |
| | puhelinnumero: | koulu: | aloituspäivä: | lopetuspäivä: |
| | <input type="text"/> | Turun amk ▾ | <input type="text" value="2011-09-09"/> | <input type="text"/> |
| | tuntilista: yhteensä | tuntien lisäys: | pvm: | tunnit: |
| | | <input type="text"/> | <input type="text"/> | <input type="button" value="ok"/> |
| | <input type="button" value="tallenna"/> | <input type="button" value="asiakastapahtumat"/> | <input type="button" value="huollot"/> | |
| aika | kesto | tyyppi | | |
| 2011-09-12 | 5 | vastaanotto | | |
| 2011-09-13 | 5 | vastaanotto | | |

Kuvassa työntekijöiden muokkauslomake.

- Yläosassa näkyy uusi työntekijä –lomakkeella annetut tiedot, joita voi tarvittaessa muokata.
- Näiden tietojen alla on listaus työntekijän tekemistä tunneista ja näiden lisäyslomake.
- Asiakastapahtumat- ja huollot-napeista saa listauksen tämän työntekijän toimimisesta.

Kansalaisen Mikrotuki

| | | | |
|--|---|----------------------|----------------------|
| Etusivu | etunimi: | sukunimi: | puhelinnumero: |
| Vastaanotto | <input type="text"/> | <input type="text"/> | <input type="text"/> |
| Muu asiakastapahtuma | koulu: | aloituspäivä: | |
| Koneen muokkaus | Turun amk ▾ | <input type="text"/> | |
| Työntekijöiden tarkastelu | <input type="button" value="tallenna"/> | | |
| Uusi työntekijä Ajanvaraus | | | |

Kuvassa uusi työntekijä -lomake. Lomakkeeseen täytetään työntekijän tiedot, koulu valitaan projektipäälliköiden tallentamista vaihtoehdoista.

Kansalaisen Mikrotuki

[Etusivu](#)
[Vastaanotto](#)
[Muu asiakastapahtuma](#)
[Koneen muokkaus](#)
[Työntekijöiden tarkastelu](#)
[Uusi työntekijä](#)
[Ajanvaraus](#)

« September 2011 »

| Mon | Tue | Wed | Thu | Fri | Sat | Sun |
|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|
| | | | 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| 26 | 27 | 28 | 29 | 30 | | |

Lisää uusi aika päivälle 2011-09-09

nimi **puhelinumero**

aika **arvioitu kesto**

Varatun ajan aihe

Varauksen vastuhenkilö

▼

tallenna

nimi puhelinnumero Varatun ajan aihe

Kuvassa ajanvarauslomake.

- Ylhäällä olevassa kalenterissa näkyy valitun kuukauden päivät. Oletuksena nykyinen kuukausi, jolloin nykyinen päivä on punainen.
- Arvioitu kesto on tekstimuodossa, joten sen voi täyttää haluamallaan tavalla. Esim. noin tunti, 30 min, 3600 sek.
- Vastuhenkilö valitaan aktiivisista työntekijöistä.
- Päivälle valmiiksi varatut ajat näkyvät lomakkeen alareunassa, esimerkiksi kuvassa niitä ei ole.

| etunimi | sukunimi | aloituspäivä | lopetuspäivä |
|---------|----------|--------------|--------------|
| M | A | 2011-02-14 | 2011-03-11 |
| M | H | 2011-03-22 | 2011-05-27 |
| T | H | 2011-01-25 | 2011-06-01 |
| J | H | 2009-05-01 | 2011-04-04 |
| S | H | 2011-04-13 | 2011-04-15 |
| J | H | 2011-05-09 | 2011-05-11 |
| M | J | 2011-04-27 | 2011-04-29 |
| P | K | 2011-02-28 | 1970-01-01 |
| L | L | 2011-05-09 | 2011-05-11 |
| V | L | 2011-04-27 | 2011-04-29 |
| L | L | 2011-01-31 | 2011-03-11 |
| A | M | 2011-01-01 | 1970-01-01 |
| J | M | 2011-01-31 | 2011-02-04 |
| J | N | 2011-04-13 | 2011-04-15 |
| M | O | 2010-09-15 | 1970-01-01 |
| G | O | 2011-04-18 | 2011-05-28 |
| G | O | 2011-09-05 | |
| H | P | 2010-09-15 | 2011-03-31 |
| R | P | 2011-08-29 | 1970-01-01 |
| L | R | 2011-02-14 | 2011-02-15 |
| m | s | 2011-02-10 | |
| Il | S | 2008-10-10 | 2011-04-17 |
| S | S | 2011-08-22 | 1970-01-01 |

Kuvassa työntekijälistaus. Projektipäällikkö tarvitsee työssään työntekijälistaus-
ta, joten järjestelmässä on sellainen. Listaus on tarkoitettu ainoastaan projekti-
päällikön nähtäville, joten siihen ei ole linkkiä, vaan se on saatavilla KMT:n ver-
kossa osoitteessa <http://192.168.1.200/uusi/tyontekijalistaus.php>

käyttöjärjestelmä:

Lisää käyttöjärjestelmä

tapahtumatyyppi:

Lisää tapahtumatyyppi

konetyyppi:

Lisää konetyyppi

työvaihe:

Lisää vaihe

koulu:

Lisää koulu

Kuvassa valintojenlisäyslomake. Esimerkiksi uusia käyttöjärjestelmiä tulee jatkuvasti, joten näiden lisäämiselle on tarve. Lomake on tarkoitettu ainoastaan projektipäällikön nähtäville, joten siihen ei ole linkkiä, vaan se on saatavilla KMT:n verkossa osoitteessa <http://192.168.1.200/uusi/admin.php>