LAPPEENRANNAN TEKNILLINEN YLIOPISTO
Tietotekniikan osasto
Tietoliikennetekniikan laboratorio

Rakennusprojektin laadunvarmistuksen automatisointi mobiiliteknologialla

Diplomityön ohjaaja: Harri Hämäläinen
Diplomityön tarkastajat: Jouni Ikonen, Jari Porras

Lappeenrannassa 16.3.2008

Teemu Reisbacka
Teknologiapuistonkatu 2 D 59
53850 Lappeenranta
Puh. +358407454362

TIIVISTELMÄ

Tekijä: Teemu Reisbacka

Työn nimi: **Rakennusprojektin laadunvarmistuksen automatisointi mobiiliteknologialla**

Osasto: Tietotekniikan osasto

Vuosi: 2008

Paikka: Lappeenranta

Diplomityö. Lappeenrannan teknillinen yliopisto. 75 sivua, 20 kuvaa, 4 taulukkoa ja 4 liitettä.

Tarkastajat: Dosentti Jouni Ikonen, TkT Jari Porras

Hakusanat: SOAP, Web Service, tietojärjestelmät, RFID

Rakennusprojekteissa yksi haastava osa-alue on laadunvarmistus: Suomen elementtitehtailla se tapahtuu tällä hetkellä käsityöllä, eikä automaatiota käytetä. Lappeenrannan teknillisen yliopiston Mobilding-hankkeessa rakennuselementteihin upotetaan radiotunnisteita, joiden avulla elementit voidaan tunnistaa langattomasti ja yksilöllisesti, sekä yhdistää tietojärjestelmän tietoon. Käyttäen hyväksi kykyä tunnistaa elementit sähköisesti, tässä diplomityössä keskitytään ratkaisemaan laadunvarmistuksen haastetta automatisoimalla prosessia.

Työssä kartoitetaan laadunvarmistuksen nykytila rakennusteollisuudessa ja sen pohjalta suunnitellaan ja tuotetaan laadunvarmistusjärjestelmä. Toteutettava järjestelmä kykenee havainnoimaan poikkeuksia reaktiona käyttäjien syötteeseen ja valvomaan projektin aikataulutusta käyttäen hyväksi elementtien tilatietoja. Havaituista poikkeuksista tiedotetaan automaattisesti. Järjestelmään toteutetaan rajapinta Web Service-teknologioilla, jolloin sitä voidaan käyttää matkapuhelimella. Työn tuloksena saatavaa järjestelmää testataan pilottihankkeissa ja siitä saadaan pohja laadunvarmistuksen jatkokehitykselle.

ABSTRACT

Author: Teemu Reisbacka

Title: **Automating the Quality Assurance Process of a Construction Project with Mobile Technology**

Department: Department of Information Technology

Year: 2008

Place: Lappeenranta

Diploma thesis. Lappeenranta University of Technology. 75 pages, 20 pictures, 4 tables and 4 appendices.

Supervisor: Docent Jouni Ikonen, Dr. Tech. Jari Porras

Keywords: SOAP, Web Service, information systems, RFID

One of the challenges in a construction project is quality assurance: in Finland it is mostly done manually without automation. In the Mobilding-project of Lappeenranta University of Technology concrete construction elements were embedded with RFID-tags, making it possible to identify them wirelessly and associate them to data in an information system. Taking advantage of this ability to electronically identify the elements, this thesis concentrates on solving the challenge by automating the quality assurance process.

The thesis charts what is the current state of quality assurance in construction industry and based on that, a solution automating the process will be designed and implemented. The implementation will be a quality assurance system capable of detecting and notifying of errors. Errors can be detected from user input or by monitoring the project scheduling and the state information of construction elements. A Web Service interfaces to the system will also be implemented, enabling communication with mobile devices. The system, which is the result of the thesis, will be tested in pilot-projects and will function as basis for future development in quality assurance.

LYHENTEET

CORBA	Common Object Request Broker Architecture
CSS	Cascading Style Sheets
CSV	Comma-separated values
DCOM	Distributed Component Object Model
DIME	Direct Internet Message Encapsulation
EJB	Enterprise JavaBean
PEAR	(PHP Extension and Application Repository
PHP	PHP: Hypertext Preprocessor
MIME	Multipurpose Internet Mail Extensions
MSC	Message Sequence Chart
MVC	Model View Controller
HTTP	Hypertext Transfer Protocol
RFID	Radio Frequency Identification
RMI	Remote Method Invocation
RPC	Remote Procedure Call
SMS	Short Message Service
SOAP	Simple Object Access Protocol
UDA	User Defined Attributes
UDDI	Universal Discription, Discovery and Integration
URL	Uniform Resource Locator
WSDL	Web Services Description Language
XML	Extensible Markup Language

SISÄLLYSLUETTELO

1. JOHDANTO

Rakennustyömaan hallinta on usein kaaoksen hallintaa. Aikataulut eivät aina pidä paikkaansa ja tarkan tiedon välittäminen eri osapuolten välillä on ongelmallista. Jos tietoa ei saada jaettua tehokkaasti, vaikeuttaa tämä esimerkiksi aikataulutusta ja poikkeustilojen hallintaa. Tämä diplomityö on osa Mobilding-hanketta, jossa selvitetään millaisilla menetelmillä rakennukseen liittyviä tietoja voidaan hallita sen elinkaaren ajalta; alkaen suunnittelusta, jatkuen rakennukseen ja käyttöönottoon. Hankkeessa sovelletaan tietotekniikkaa ratkaisemaan rakennusteollisuuden ongelmatilanteita ja parantamaan tehokkuutta. Hankkeen ytimenä on käyttää RFID-tunnisteita (Radio Frequency Identification) tunnistamaan rakennuselementit yksilöllisesti, jolloin ne voidaan yhdistää tietojärjestelmän tietoon.

Laadunvarmistus on yksi ongelmista, joihin törmätään rakennusteollisuudessa. Eri osapuolille laatu merkitsee eri asioita. Valmistajan näkökulmasta ongelmana on varmistaminen, että kaikki rakennustyömaalle lähetettävät elementit täyttävät niille asetetut vaatimukset. He haluavat varmistaa, että elementit lähtevät asiakkaalle virheettöminä ja aikataulussa. Elementtien täytyy vastata mittasuhteiltaan ja ulkoiselta rakenteeltaan suunnitelmia. Jos lähetetyt elementit ovat virheettömiä, voi valmistaja minimoida ylimääräiset kustannukset, kuten reklamaatiot ja elementtien korjaukset. Rakennustyömaan näkökulmasta ongelma on virheistä ilmoittaminen takaisin valmistajalle; esimerkiksi voidaanko havaittu virhe korjata vai joudutaanko elementtejä vaihtamaan. Ongelmatilanteet tulisi havaita mahdollisimman ajoissa ja niistä tulisi ilmoittaa osapuolille mahdollisimman nopeasti, jotta aiheutuneet kustannukset saataisiin minimoitua. Myös käytetyllä ajalla on tässä tapauksessa yhteys rakennusprosessin laatuun: pahimmassa tapauksessa virhe tai puuttuva elementti voi aiheuttaa rakentamisen pysähtymisen, joka aiheuttaa suuren määrän ylimääräisiä kuluja. Tässä diplomityössä pohditaan tätä laadunvarmistusongelmaa tietojärjestelmän kannalta ja toteutetaan laadunvarmistusjärjestelmä, jolla prosessia voidaan automatisoida.

Työssä ongelmaa lähdetään ratkaisemaan selvittämällä millainen nykytilanne

rakennusteollisuudessa on. Oleellisimmiksi kysymyksiksi nousevat:

- Millaisia virheitä tai poikkeamia on olemassa eri vaiheissa?
- Miten näihin reagoidaan?
- Ketä ja miten virheistä ja poikkeamista informoidaan?

Tapahtumaketjun kartoituksen pohjalta tämän diplomityön tuloksena suunnitellaan ja toteutetaan laadunvarmistusjärjestelmä sekä liittymä- ja rajapinnat tähän järjestelmään. Toteutusta tullaan testaamaan käytännön olosuhteissa ja tulosten muodostavan pohjan perusteella voidaan tehdä jatkokehitystä. Pilottihankkeissa yhteystyökumppaneina on suomalaisia elementtitehtaita, joten toteutus käyttöliittymissä tulee keskittymään enemmän heidän tarpeidensa ympärille.

Tämä diplomityö muodostaa kokonaisuuden Teemu Vilkon diplomityön "Mobiiliteknologia liikkuvan työntekijän apuna" /VIL07/ kanssa, jossa on toteutettu matkapuhelinsovellus tietojen välittämiseen puhelimen ja tietojärjestelmän välillä. Sovelluksen avulla voidaan esimerkiksi rakennustyömaalla tai elementtitehtaan työalueilla tunnistaa langattomasti RFID-tunnisteen sisältävät rakennuselementit ja välittää tai hakea näihin liittyvää tietoa. Kommunikoinnin mahdollistava rajapinta toteutetaan Web Service-teknologioilla, sillä aikaisemmin toteutetussa kandidaatintyössä ”Web Servicen käyttö rajapintana tiedonsiirrossa” /REI07/ se oli havaittu hyväksi vaihtoehdoksi. Saman työn tuloksena on mahdollistettu Tekla Structures-rakennusmallien tietojen synkronisoiminen tietojärjestelmän kanssa. Nämä kolme työtä muodostavat kokonaisuuden laadunvarmennuksessa, mahdollistaen automaattisen rakennussuunnitelmien tuomisen järjestelmään ja tehokkaan tavan käyttää sitä.

2. NYKYTILANNE JA RFID:N KÄYTTÖ RAKENNUSTEOLLISUUDESSA

Mobilding-hanke pohjautuu RFID-teknologian käyttöön ja radiotunnisteiden upottamiseen elementteihin jo valmistusvaiheessa. Rakennusteollisuudessa tällä voidaan saavuttaa elementtien tai tuotteiden tunnistaminen etäluvun avulla. Elementit voidaan tunnistaa langattomasti etäisyyden riippuessa materiaalista, johon tunniste on upotettu. Verrattuna esimerkiksi viivakoodeihin, radiotunnisteet pysyvät suojattuina elementtien sisällä koko elinikänsä ajan, jolloin tunnistaminen on mahdollista vielä vuosien päästä. Luettu, yksilöllinen tunnus voidaan yhdistää tietojärjestelmän tietoon ja näin voidaan automatisoida prosesseja; oli se sitten tietojen siirtämistä, elementtien tilojen seurantaa tai virheistä ilmoittamista. Ennen kuin lähdemme suunnittelemaan ratkaisuja, on meidän tiedettävä millainen tilanne on rakennustyömaalla ja elementtitehtailla tällä hetkellä. Näin voimme tuottaa hyödyllisen ratkaisun, jolla voidaan saavuttaa aika- ja kustannussäästöjä.

Nykytilanteen lisäksi on hyödyllistä tietää millaista tutkimusta samalla sovellusalueella on tehty aikaisemmin. RFID-teknologian soveltamista rakennusteollisuuden tarpeisiin on tutkittu sekä Suomessa, että Suomen ulkopuolella ja RFID-teknologian käyttö vaikuttaa olevan lisääntymässä. Mobilding-projektin kaltaisia pilottihankkeita on tehty myös muualla ja toteutettuja pilottihankkeita tarkastellessa käy ilmi, että yksi yleisimmistä kohteista on RFID-teknologian soveltaminen ihmisten tarkkailuun ja kulunvalvontaan rakennustyömailla. Myös elementtien etätunnistaminen on ollut tutkimuksen kohteena aikaisemmin. Tässä luvussa käydään aluksi läpi suomalaisen rakennusteollisuuden nykytila, jonka jälkeen esitellään lyhyesti millaisia Mobilding-projektin kaltaisia pilottihankkeita on tehty muualla.

2.1 Nykytilanteen kartoitus

Laadunvarmistuksen nykytilannetta rakennusteollisuudessa kartoitettiin haastattelemalla

rakennusteollisuuden edustajia, erityisesti elementtien valmistajia /SUI07, KAU07, KOS07/. Haastatteluista kävi ilmi, että tällä hetkellä suuri osa laadunhallinnasta tapahtuu käsityöllä, ilman mitään automaatiota. Valmistajat hoitavat laadunvarmistusta sisäisesti tehtaalla ja luonnollisesti reagoivat saatuihin virheilmoituksiin.

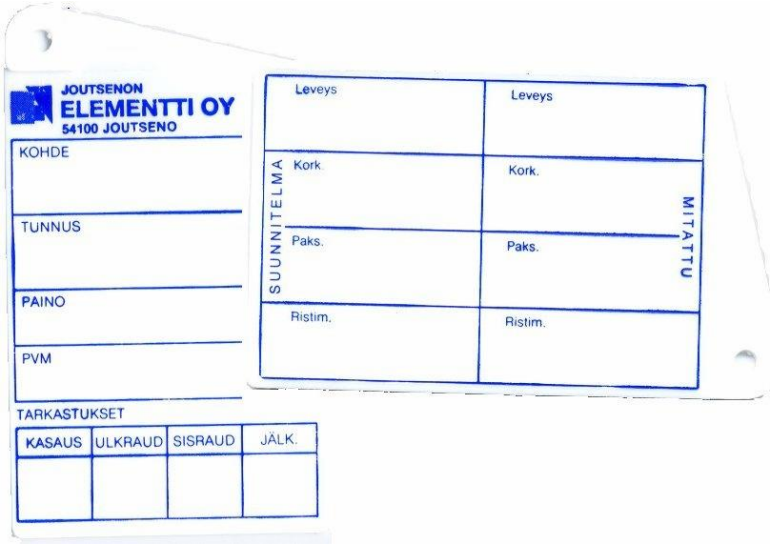
Päätökset poikkeamiin reagoimisesta tehdään pääsääntöisesti ihmisten ammattitaidon pohjalta ja ne tehdään yleensä elementtien valmistajan puolella. Tästä syystä on tärkeää, että rakennustyömaa ilmoittaa virheistä valmistajalla mahdollisimman tarkasti ja nopeasti. Pahimmassa tapauksessa esimerkiksi vioittunut rakennuselementti voi estää rakentamisen jatkamisen ja kaikki osapuolet pysähtyvät odottamaan korvaavaa elementtiä, jolloin syntyvät lisäkustannukset ovat suuret. Seuraavat kappaleet kuvailevat, kuinka laadunvarmistus toimii tällä hetkellä haastatelluissa tehtaissa ja kuinka kommunikaatio osapuolten välillä tapahtuu.

2.1.1 Elementtien tarkistus tehtaalla

Elementtejä tarkistetaan tehtaalla sekä pintapuolisesti, että mittaamalla. Silmin havaittavia puutteita ovat esimerkiksi fyysiset vauriot tai värivirheet elementeissä /SUI07/. Näiden lisäksi tarkastetaan valmistuneiden elementtien dimensioita mittaamalla niitä ja vertaamalla mittauksia suunniteltuihin dimensioihin. Elementtien on täytettävä laatustandardit, jotta ne voidaan lähettää asiakkaalle. Viallisten elementtien lähettäminen aiheuttaa suuria lisäkustannuksia valmistajalle: kun rakennustyömaa tekee reklamaation, aiheutuu siitä lisäkustannuksia kuljetusten, työtuntien ja materiaalin muodossa.

Laadunvalvontaa suoritetaan vaihtelevalla tiheydellä. Suurimpana esteenä elementtien tarkistamiselle vaikuttaa olevan tarkistamiseen tarvittava työmäärä. Elementtien mittaukset ja tarkastukset hoidetaan tehtailla käsin ja se sitoo työntekijöitä ja aikaa toimenpiteen suorittamiseen. Tämä aiheuttaa tehtaalle lisäkustannuksia, joita yritetään välttää. Osassa elementtitehtaita tarkastukset tehdään satunnaistarkistuksina: valmistuvasta erästä tarkistetaan vain tietty osa ja varmistetaan, että tarkastetut elementit ovat toleranssien sisällä /KAU07/. Jos asiakas kuitenkin vaatii tarkastuksia, jokainen elementti tarkastetaan

erikseen. Tämä kuitenkin aiheuttaa ylimääräisiä kuluja, eikä sitä tehdä kaikille tilauksille automaattisesti.



Kuva 2.1. Joutsenon Elementti Oy:n tarkastuslaput

Osassa elementtitehtaista mittatarkastuksia on viety hieman pidemmälle. Elementtien valmistustulosteisiin on jätetty tyhjät kentät dimensioille, joihin työntekijät merkitsevät elementin valmistuneet mitat. Mittaukset tehdään esimerkiksi Parma Oy:ssä käsin ja tarkistuslaput kootaan jälkeenpäin yhteen. Myös Joutsenon Elementti Oy käyttää tarkistuksissa elementteihin kiinnitettäviä lappuja, joihin mitatut dimensiot kirjoitetaan. Esimerkki tarkastuslapuista on nähtävissä kuvasta 2.1.

Vaikka elementeistä tehdään mittaukset, tietoa ei välttämättä tallenneta keskitetysti mihinkään. Mittausten tarkoitus on todeta valmistushetkellä, voidaanko elementti lähettää eteenpäin. Tämän jälkeen tulokset kuitenkin hävitetään tai arkistoidaan paperimuodossa mappeihin /KAU07, KOS07/. Jälkeenpäin on siis mahdotonta tai erittäin hankalaa kerätä mitään tilastotietoa, minkälaisia laatuvaihteluita tehtaalla on tapahtunut ja hyödyntää kerättyjä tuloksia.

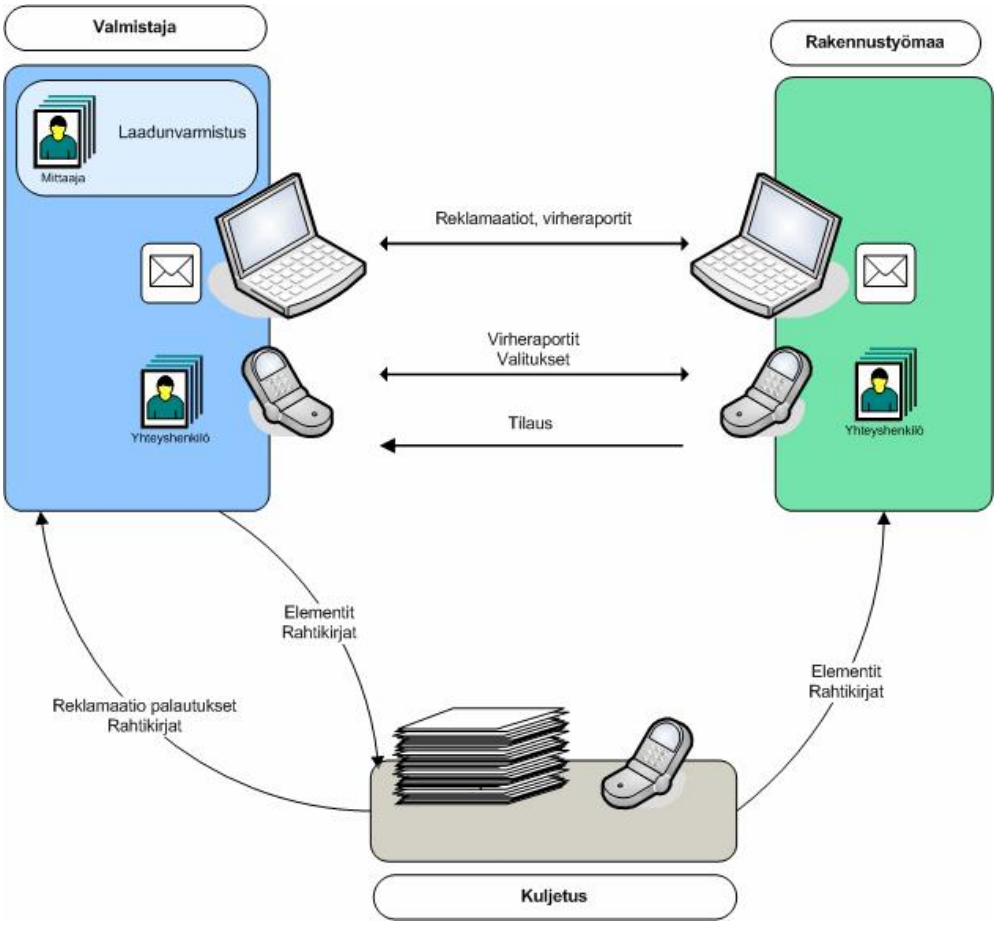
Virheisiin reagoinnissa ei myöskään ole monia kirjoitettuja standardeja, vaan päätökset

tehdään työntekijöiden ammattitaidon perusteella /KOS07/. Betonielementeillä on tietyt toleranssirajat, jotka rakennusteollisuus on määrittänyt. Jos elementti eroaa liikaa näistä, on se hylättävä. Päätökset, milloin elementti on mahdollista korjata ja lähettää asiakkaalle, tehdään tilannekohtaisesti.

2.1.2 Virheiden havainnointi rakennustyömaalla

Virheistä ilmoitetaan rakennustyömaalla lähinnä kahdessa vaiheessa: elementtejä asennettaessa ja lopputarkastusta tehdessä /SUI07/. Jos elementissä on selkeä vika (värivirhe, selkeä rakennevirhe), se voidaan havaita rakennustyömaalla jo suoraan toimituksen vastaanotossa. Muussa tapauksessa suuremmat virheet havaitaan vasta asennuksen yhteydessä, kun huomataan, ettei elementti sovi sille suunniteltuun paikkaan. Tämä on kuitenkin jo liian myöhäistä, kun ajattelee jo pelkästään aiheutuvia kustannuksia ja virheet tulisi havaita aikaisemmassa vaiheessa. Pienemmät virheet katselmoidaan vasta rakennusprojektin lopussa kootusti, eikä niistä ilmoiteta tapauskohtaisesti.

Haastattelujen pohjalta kommunikointisuhteita on esitelty kuvassa 2.2. Yleisimpinä kommunikaatiovälineinä poikkeustilanteissa toimii puhelin, mutta myös sähköpostia käytetään. Rakennustyömaa ottaa yleensä yhteyttä valmistajaan, joka päättää toimenpiteistä ja on yhteydessä kuljetusyritykseen tarpeen mukaan. Rakennustyömaa ei itse keskustele kuljetusyrityksen kanssa, vaan tämä viestimien on valmistajan vastuulla. Seuraavat kappaleet käyvät läpi näitä kommunikointisuhteita, virheiden havainnointia ja niistä tiedottamista yksityiskohtaisemmin.



Kuva 2.2. Rakennusprojektin kommunikointi käytännössä

2.1.3 Asennuksessa havaittavat poikkeamat

Suurin osa virheistä, poislukien aivan ilmeiset ongelmat elementeissä, havaitaan vasta asennuksen yhteydessä /KA07B/. Tällöin asentaja tai vastuuhenkilö rakennustyömaalla ottaa yhteyttä valmistajaan. Virhekuvauksen perusteella valmistaja päättää korjaavista toimenpiteistä; pahimmassa tapauksessa tämä saattaa tarkoittaa elementin hylkäämistä ja uuden valmistamista, mutta myös korjaaminen on mahdollista /KOS07/. Tällöin valmistaja lähettää korjaajan rakennusmaalle.

Yleisimmät virheet, mihin asennuksessa törmätään, ovat mittavirheitä /KOS07/. Elementit

ovat liian pitkiä tai liian lyhyitä ja tästä johtuen asennettavassa kokonaisuudessa tapahtuu heittoja. Elementti ei saata enää mahtua sille tarkoitetulle paikalle tai kokonaisuuteen jää liian suuria tyhjiä aukkoja. Parma Oy:n Mikko Koskisen /KOS07/ mukaan muita yleisiä virheitä ovat osittaiset hajoamiset, sekä virheet laattojen ja palkkien varauksien ja varusteluiden paikoissa ja koossa. Varauksilla ja varusteluilla tarkoitetaan, että samantyyppisissäkin elementeissä voi olla eri paikoissa asennukseen tarvittavia reikiä tai muutoksia rakenteessa.

Myöskään näissä virheissä ei ole tehtaiden kannalta sovittuja käytäntöjä siitä, kuinka virheisiin reagoidaan. Päätökset tehdään kokemuksen perusteella. Jos elementti on havaittu esimerkiksi liian pitkäksi, voidaan tehtaalta lähettää työntekijä korjaamaan ongelmaa. Toimenpiteet ovat kuitenkin tapauskohtaisia.

2.1.4 Jälkikatselmointi

Osa virheistä käsitellään kootusti jälkikatselmuksessa. Tänne jätetään havaitut virheet, joiden ei ole katsottu haittaavan rakentamista. Koska virheet eivät ole kriittisiä, niistä ei ilmoiteta valmistajalle havaitsemisen yhteydessä. Ne kirjataan ylös ja käsitellään yhdessä, kun rakennusprojekti valmistuu.

Katselmoinnin tarkoituksena on lähinnä päättää, kuka joutuu maksamaan näiden virheiden korjauksesta. Vastuu saattaa olla elementin valmistajan tai rakennustyömaan. Tällaisista virheistä ei ilmoiteta etukäteen, sillä syntyvän ylimääräisen työn ja kommunikaation määrä ei ole suhteessa kustannuksiin. Jos teknologia kuitenkin tekee viestimisen kyllin helpoksi, tieto olisi hyvä saada välitettyä mahdollisimman aikaisessa vaiheessa /SUI07/. Näin valmistajakin voisi valmistautua etukäteen ja tietäisi tilanteen virheistä jatkuvasti.

2.1.5 Kommunikointi ja yhteyshenkilöt

Kun virhe havaitaan rakennustyömaalla, tapahtuneesta ilmoitetaan soittamalla suoraan

valmistajan tehtaalle. Valmistaja päättää tämän jälkeen jatkotoimenpiteistä ja tiedottaa tästä rakennustyömaata /KOS07/. Usein raportti lähetetään myös sähköpostilla. Raportointiin on hyvä saada ns. paperivarmennus, jolloin raportti on arkistoitavassa muodossa ja voidaan olla varmempia, että mitään yksityiskohtia ei unohdettu kertoa /KA07B/.

Raportointi virheestä tapahtuu sanallisessa muodossa ja se on usein hyvin lyhyt kuvaus. Asentaja voi esimerkiksi ilmoittaa, että elementti on havaittu liian pitkäksi tai liian lyhyeksi eikä sovi suunniteltuun paikkaan. Haastatellut rakennusteollisuuden edustajat ilmaisivat, että kuvien lähettäminen virheen raportoinnin yhteydessä olisi tarpeellinen ominaisuus. Koska päätöksen jatkotoimenpiteistä tekee valmistaja, jolla ei välttämättä ole edustajia rakennustyömaalla, on tärkeää, että he saavat mahdollisimman tarkan kuvauksen ja mahdollisimman paljon tietoa virheestä.

Rakennustyömaan ja valmistajan välisessä kommunikoinnissa molemmalla osapuolilla on yhteyshenkilö. Kun valmistaja vastaanottaa tilauksen, tilaajalle/rakennustyömaalle annetaan yrityksessä yhteyshenkilö, joka on vastuussa viestimisestä. Vastaavasti tilaaja ilmoittaa oman yhteyshenkilönsä tilauksen yhteydessä. Jos esimerkiksi asennuksen yhteydessä havaitaan elementissä virhe, voidaan rakennustyömaalta soittaa suoraan henkilölle, joka on asiasta vastuussa. Pienemmillä asiakkailla ei ole suoraa yhteyshenkilöä kenelle he voivat soittaa, vaan ongelmatilanteessa he ottavat yhteyttä tehtaaseen, josta heidät ohjataan henkilölle, joka käsittelee ongelman.

2.2 Toteutettuja pilottihankkeita Suomessa

Mobilding-hanke ei ole ainoa, jossa radiotunnisteiden käyttöä on testattu rakennusteollisuuden käyttötarkoituksissa. RFID-teknologian käyttö Suomen rakennusteollisuudessa on vielä kokeiluasteella, mutta esimerkiksi mobiiliteknologiaa on jo käytössä. Sonja Leskinen on kartoittanut gradutyössään ”Mobile Solutions and Construction Industry – Is it a working Combination?” /LES06/ Suomen rakennusteollisuuden käyttöön toteutettuja mobiilisovelluskokeiluja.

Yleisesti RFID-teknologian käyttöä rakennusteollisuuden apuna on tutkittu TeliaSoneran kanssa yhteistyössä tehdyssä Jobsite Logistics-hankkeessa, jossa virtuaalimallin tietoa linkitettiin reaalimaailman elementteihin RFID-tunnisteiden avulla. Perusidea oli hyvin samankaltainen kuin Mobilding-hankkeessa. Tarkoituksena tässä pilottihankkeessa oli logistiikan tehostaminen. Tämän diplomityön aihealueeseen liittyy myös Buildercom:n työturvallisuuden valvonta-pilotti. Siinä radiotunnisteita ei yhdistetty rakennuselementteihin, vaan ajatuksena oli testata virheraportointia langattomasti mobiililaitteilla.

2.2.1 Jobsite Logistics

Jobsite Logistics hanke toteutettiin elokuussa 2005 yhteistyönä Skanskan, Nokian, Fonestra Oy:n, RKL A Taskinen Oy:n ja Enterprixe:n välillä. Projektissa keskityttiin logistiikan parantamiseen rakennustyömaalla. Tutkimusongelma hankkeessa oli virtuaalimallin yhdistäminen reaalimaailmaan ja fyysisiin elementteihin reaaliajassa. Tässä on selkeitä yhteneväisyyksiä Mobilding-hankkeessa toteutettuun pilottiin, jossa Teklan Tekla Structures-mallista tuotiin tietoa Mobilding tietojärjestelmään ja yhdistettiin se reaalimaailman objekteihin.

Rakennustyömaan työnjohtaja saattaa päivittäin joutua tekemään jopa kaksi tuntia logistiikkaan liittyvää paperityötä. Jobsite Logistics-hankkeessa tästä virhealttiista mekaanisesta tarkastustyöstä haluttiin päästä irti ja se haluttiin automatisoida. Tämä ratkaistiin pilottihankkeessa liittämällä RFID-tunniste osaan elementeistä. Nämä RFID-elementit oli liitetty Enterprixen malliin rakennustyömaasta ja RFID-tunnisteita seurattiin koko prosessin läpi: suunnittelussa, kuljetuksessa, rakennustyömaan vastaanotossa ja asennuksessa. RFID-tunnisteet luettiin prosessin eri vaiheissa Nokian 5140i puhelimella, jossa oli RFID-lukija.

Pilottihanke vietiin loppuun ja se koettiin onnistuneeksi, koska se lisäsi logistiikkaprosessien läpinäkyvyyttä kaikille osapuolille. Pilotista tehtiin seuraavanlaisia johtopäätöksiä:

- Tiedon kulku oli tarkkaa ja avusti pitämään projektin aikataulussa
- Oikeat elementit olivat oikeassa paikassa oikeaan aikaan, mikä auttoi säästämään työkustannuksissa halki koko projektin
- Valittu Nokia 5140i-puhelin havaittiin helpoksi käyttää ja se toimi hyvin sekä tehdas-, että rakennustyömaaolosuhteissa
- RFID-lukija toimi oletetulla tavalla ilman ongelmia

Projektin jälkeen osapuolten suurin toivomus oli ominaisuus, että kaikki RFID-tunnisteet voitaisiin lukea suoraan kuormasta ”pyyhkäisemällä”. Johtopäätöksissä todettiin, että tämä saattaa olla mahdollista käyttämällä kahta RFID-tunnistetta; yksi tunniste elementille ja yksi tunniste kuormakokonaisuudelle. Tällöin oletettaisiin, että koko kuormaava kuvaava tunniste toimisi lähetteenä, eikä olisi virheellinen.

2.2.2 Turvallisuuden valvonta

Buildercom:n pilottihanke aloitettiin vuonna 2005 ja sen tarkoituksena on automatisoida turvallisuustarkistuksia. Suomessa valtioneuvosto vaatii, että jokainen rakennustyömaa valvoo rakennusturvallisuutta säännöllisesti /VAL94/. Valvonta tapahtuu siten, että tarkastaja kiertää rakennustyömaata, ja dokumentoi näkemänsä tiettyjen laatukriteerien mukaan. Rakennustyömaa katsotaan turvalliseksi, jos 85 % havainnoista on virheettömiä.

Buildercom:n hankkeessa tämä prosessi siirretään Nokian S60-sarjan matkapuhelimeen. Tarkastaja merkkaa huomatuksensa puhelimeen ja voi samalla myös ottaa puhelimen kameralla kuvan virheestä. Tämän jälkeen puhelin lähettää raportin joko Buildercom:n järjestelmään tai mahdollisesti suoraan virheestä vastuussa olevalle henkilölle (suoraan / sähköpostilla). Järjestelmään syötettäessä virheiden tilaa voidaan tarkkailla jatkuvasti Internetin välityksellä.

Normaalisti kaikki tämä dokumentaatio tehdään paperille. Tämä on kuitenkin aikaavievää

ja paperit voivat kadota, joten hankkeella yritetään saavuttaa seuraavia hyötyjä:

- Halutaan lisää varmuutta dokumentointiin
- Halutaan lisää nopeutta

Projektin on ollut tarkoitus loppua vuoden 2006 lopussa /BUI05/, mutta loppuraporttia ei ollut kirjoitushetkellä vielä saatavilla.

2.3 Hankkeita Suomen ulkopuolella

RFID-teknologian käyttöä rakennusteollisuudessa on tutkittu myös Suomen ulkopuolella. Suomen Tekesin, Tanskan Danish Enterprise and Construction Authorityn ja ruotsalaisen Swedish Research Council for Environment, Agricultural Sciences and Spatial Planning -järjestön kanssa toteutetun Era Build -projektin loppuraportissa ”RFID in Construction” /ERA06/ on laajalti selvitetty RFID:n käyttöä rakennustyömailla ja rakennusteollisuudessa ympäri maailmaa. Raportissa kuvattujen toteutettujen hankkeiden puolesta suurimmat kiinnostuksen kohteet näyttävät olevan henkilöstön valvonta RFID:n avulla, mutta myös RFID-tunnisteiden valaminen tai liittäminen rakennuselementteihin on havaittu kiinnostavaksi. Molemmissa tapauksissa motivaatio on tunnistamisella; elementti tai ihminen saadaan tunnistettu yksilöllisesti ja kohteesta saadaan nopeasti yksityiskohtaista tietoa.

2.3.1 Radiotunniste resurssien valvonnassa

Raportissa esitellyistä hankkeista suurin osa liittyi resurssien valvontaan, oli se sitten kulunvalvontaa tai laitteiston valvontaa. Sekä USA:sta, Kanadasta, että Euroopasta löytyy useita toteutettuja pilottihankkeita, joissa RFID-teknologiaa on rakennusteollisuudessa käytetty tähän tarkoitukseen. Piloteissa on pääsääntöisesti keskitytty seuraamaan tunnisteiden avulla joko rakennustyömaan työntekijöitä tai rakennetyömaalle tuotavia materiaaleja ja kalustoa.

Suuri syy työntekijöiden tarkkailuun on kulkuoikeuksien määrittäminen ja varkauksien vähentäminen rakennustyömaalla. Esimerkiksi Kanadassa National Equipment Register (NER) on laskenut että rakennusteollisuuden yritykset menettivät vuonna 2005 miljardi dollaria pelkästään kadotetun tai varastetun kaluston vuoksi. Useassa pilotissa valvontaa on testattu henkilöstölle annetun RFID-tunnistekortin avulla. Esimerkkejä toteutetuista pilottihankkeita ovat mm. :

- Ruotsissa Safetool toteutti hankkeen, jossa passiivinen RFID-tunniste liitetään laitteistoon, työvälineisiin ja tunnistekortteihin. Nämä kommunikoivat aktiivisten lukijoiden kanssa.
- Ruotsissa TracTechnology toteutti GateTrac-Construction pilotin Skanskan rakennusprojektissa Tukholmassa. TracTechnology toimitti Skanskalle ratkaisun, jossa rakennustyömaan sisäänkäynneille toteutettiin elektroninen valvonta ja työntekijöille jaettavia RFID-tunnistekortteja luettiin pitkän matkan RFID-lukijoilla. Tarkoituksena oli löytää keinoja hallita ja vähentää rakennustyömaan kustannuksia.

2.3.2 Radiotunniste betoniin ja elementteihin valettuna

RFID-tunnisteiden valaminen elementteihin on aiheuttanut myös kiinnostusta maailmalla. ”RFID in Construction”-raportin mukaan motivaationa tällaisissa kokeiluissa on ollut elementistä saatava nopea, yksityiskohtainen tieto. Innovation Lab Tanskassa on toteuttanut hankkeen Intelligent Concrete, jossa betonielementtien sisään on valettu RFID-tunniste. Tämä tunniste luetaan PDA-laitteella, joka tunnisteen perusteella näyttää työmiehille heti kaiken tarvittavan tiedon elementistä: mitat, painon, tuotantohistoria ja asennus- sekä huolto-ohjeet. Saavutettavat hyödyt ovat elementin dokumentaation saatavuuden helpottaminen ja elementin seuranta.

Hong Kongissa MTR MA on toteuttanut RFID-ratkaisun betonin laaduntestauksessa. Paperitulosteiden sijasta betoninäytteiden tunnistamiseen käytetään RFID-tunnisteita. Näytteiden pintaan upotetaan tunniste betonin ollessa vielä märkää, jonka jälkeen käsilukijalla yhdistetään tunnisteen tieto elementin tietoon. Koska tunnistetta ei enää voi

poistaa elementissä hajottamatta sitä, on elementtien vaihtaminen tai korvaaminen mahdotonta. Testauksen aiheuttamien olosuhteiden takia tunnisteet on pakattu suojaavaan muovikuoreen. Tarkoituksena prosessissa on testitulosten tallentaminen elektronisesti, varmentaa tulokset ja nopeuttaa tilastollista analyysiä.

3. WEB SERVICE-TEKNOLOGIAT

Rajapinnat laadunvarmistusjärjestelmään toteutetaan tässä diplomityössä Web Servicen avulla. Suurena syynä tähän valintaan on tarve avoimuuteen ja yhteensopivuuteen, joka voidaan saavuttaa Web Service-teknologioiden avulla. Seuraavissa kappaleissa esitellään Web Service-teknologiat ja perustellaan, miksi ne soveltuvat tämän diplomityön toteutukseen erityisen hyvin. Luvussa esitellään aluksi Web Service yleisellä tasolla ja sen jälkeen pureudutaan syvemmin protokolliin ja standardeihin, joiden varaan sen toiminta perustuu.

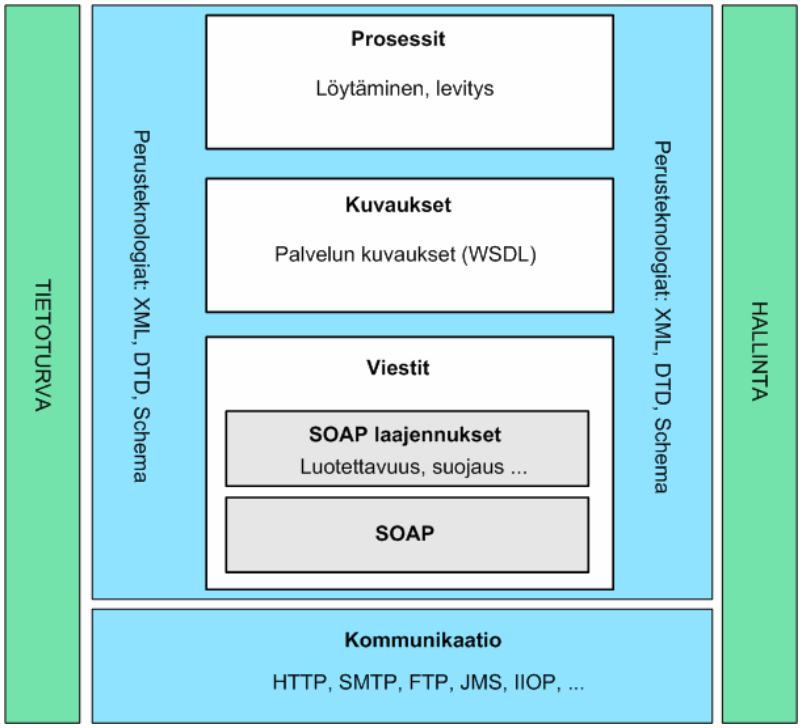
3.1 Yleiskuva

Web Servicen voi määritellä uuden sukupolven hajautetuksi tietojärjestelmäksi. Karkeasti kuvaillen Web Service on itsenäinen, itsensä kuvaava, modulaarinen sovellus, joka voidaan julkaista ja jota voidaan käyttää verkossa. Web Servicen kuvaavin ominaisuus on sen kyky peittää alleen monimutkaistakin järjestelmälogiikkaa ja funktioita ja silti tukea universaalia integraatiota ohjelmistojen kanssa /CHE06/. Teknisestä näkökulmasta Web Service koostuu kokoelmasta protokollia, jotka suorittavat ohjelmiston kuvaamisen, kommunikoinnin ja julkaisun internetissä. Web Servicen ydinajatukset ovat alustariippumattomuus ja palvelulähtöinen arkkitehtuuri /CHE06/.

Kun tarkastellaan hajautettuja tietojärjestelmiä, erilaisten monimutkaisten järjestelmien integrointi on aina ollut ongelmallista. Gustavo Alonso arvioi vuonna 2004 kirjassaan ”Web Services – Concepts, Architecture and Applications” /ALO04/, että varsinkin lyhyellä aikavälillä Web Servicet tulevat ratkaisemaan monia yhteensopivuusongelmia, jotka liittyvät ohjelmien integrointiin. Suuri ongelma on ollut standardien puute. Web Service pyrkii ratkaisemaan tätä ongelmaa perustumalla juuri palvelukohtaiseen näkökulmaan ja avoimien standardien käyttöön. Vaikka siirtymä tällaiseen uuteen arkkitehtuuriin saattaa aiheuttaa (tilapäistä) alenemista suorituskyvyssä, se tarjoaa vastapainoksi joustavuutta ja yhteensopivuutta /SER06/.

Web Service voidaan toteuttaa useammalla eri kommunikaatioprotokollalla; näistä yleisimmin käytettyjä ovat SOAP (Simple Object Access Protocol) ja XML-RPC. XML-RPC on näistä kahdesta vanhempi ja huomattavasti yksinkertaisempi. XML-RPC tähtää yksinkertaisuuteen /RPC07/, kun taas SOAP jatkaa siitä mihin XML-RPC jää, lisäten ominaisuuksia ja mahdollisuuksia määritellä viestien sisältöä ja palvelun kuvausta tarkemmin. Protokollien lähestymistavasta saa hyvän kuvan vertaamalla esimerkiksi niiden määrityksien pituutta: XML-RPC:n määritykset mahtuvat kahteen sivuun kun SOAP:n määritykset ovat noin 40 sivua /SOAP/. Konkreettisempana erona XML-RPC on kevyempänä standardina suorituskykyisempi, mutta valtaosa Web Serviceistä on silti SOAP-teknologiaan perustuvia. Se tarjoaa avoimemman ratkaisun, kuin XML-RPC ja syyt tähän käyvät ilmi kun tarkastellaan SOAP Web Servicen rakennetta.

Voimme tarkastella kuvasta 3.1 SOAP:iin pohjautuvan Web Servicen luomaa palvelua ja nähdä, kuinka se kokoaa standardeja yhteen. Web Service tarjoaa palvelun koneelta koneelle ottamatta kantaa millainen laite toisella puolella kommunikoi, kunhan se kykenee keskustelemaan tarjotuilla standardeilla. Kun alla olevaa kuvaa lähdetään käymään läpi, ylimpänä kaaviossa ovat palvelun löytämiseen tarkoitetut prosessit. Nämä prosessit eivät koske palvelun sisäistä toimintaa, vaan ne yksinkertaisesti ohjaavat asiakkaita palveluun. Ne toimivat Web Serviceiden ”keltaisina sivuina” tai hakukoneena.



Kuva 3.1. SOAP Web Services arkkitehtuuripino /W3C04/

Toisena kaaviossa on palvelunkuvaus, joka kuvailee tarjotut palvelut asiakkaalle. Sen perusteella asiakasohjelma näkee millaisia toimintoja palvelu tarjoaa ja kykenee muotoilemaan lähetyt viestit oikeaan muotoon. Palvelunkuvaus on määritetty jossain käsiteltävässä formaatissa, joka yleisimmin on WSDL (Web Services Description Language).

Kolmantena kuvassa ovat viestit, joiden avulla kommunikaatio palvelun kanssa tapahtuu. Asiakkaat keskustelevat palvelun kanssa palvelunkuvauksen määrittelemällä tavalla, lähettäen viestejä, jotka on pakattu SOAP-kuoreen. Nämä viestit puolestaan lähetetään yleensä HTTP:n (Hypertext Transfer Protocol) yli ja ne käyttävät avoimia Web-standardeja, kuten XML:ää (Extensible Markup Language) /W3C04/. Perusmuodossaan SOAP-viestit ovat rakenteeltaan yksinkertaisia, mutta viestien otsikko-kenttään voidaan liittää esimerkiksi tietoturvalaajennuksia.

Web Servicellä saadaan luotua helposti laajennettava, avoin rajapinta tai palvelu. Kyseessä voi olla esimerkiksi verkkokauppa tai tämän diplomityön tapauksessa yhteensopiva rajapinta tietojärjestelmään. Koska kommunikoinnissa käytetään avoimia, yleisesti

hyväksyttyjä standardeja, ei ole tarvetta ottaa kantaa millainen palvelun asiakas on. Päätelaite ja käyttöympäristöt ovat vapaasti valittavia ja tarjolla on avoimeen lähdekoodiin perustuvia sovelluksia.

3.2 Web Service -standardit

Kuten aiemmin on kerrottu, Web Service koostuu avoimista standardeista. Käytännössä toteutuksessa on kolme perusteknologiaa, jotka määrittelevät sen ominaisuudet. Nämä ovat kommunikaatioprotokolla SOAP, palvelunkuvauskieli WSDL, ja Web Servicen julkaisemiseen ja paikantamiseen tarkoitettu UDDI (Universal Description, Discovery and Integration). Näiden teknologioiden ymmärtäminen avaa Web Servicen toiminnallisuuden tehokkaasti ja auttaa ymmärtämään tarkalleen miten palvelu toimii.

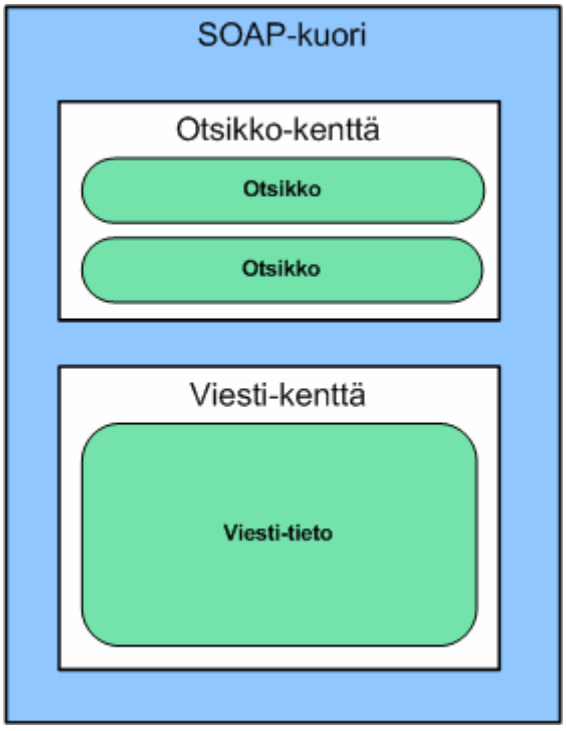
Palvelun luominen vaatii oikeastaan näistä kolmesta standardista vain kaksi ensin mainittua, SOAP:n ja WSDL:n. Nämä kaksi standardia määrittelevät kommunikaation, joka tapahtuu asiakkaan ja palvelun välillä. UDDI kuuluu palvelun löytämiseen tai julkaisuun tarkoitettuihin prosesseihin. Se on oleellinen osa julkisia palveluja, mutta sen liittäminen yksityisiin palveluihin ei ole välttämätöntä. Seuraavat kappaleet esittelevät kaikki kolme standardia yksityiskohtaisesti.

3.2.1 Tiedonsiirtoprotokolla

SOAP on protokolla XML-pohjaisen tiedon siirtämiseen. SOAP tarjoaa standardin tavan paketoida viestejä, joita kommunikoivat osapuolet lähettävät. Käytännössä SOAP onkin XML:ää; se on XML-määritysten sovellus /TID01/. Pakkaamalla tiedot SOAP-kuoreen, voidaan sopia yhteisistä viestintäsäännöistä ja määrittää käytetyt tietotyypit ja tiedon rakenne. Tämä on edellytys sille, että kaksi toisilleen tuntematonta ohjelmaa voivat keskustella toistensa kanssa.

SOAP viestin rakennetta tarkastellaan kuvassa 3.2. Viesti koostuu SOAP-kuoresta, joka sisältää valinnaisen otsikko-kentän (eng. header) ja pakollisen viesti-kentän (eng. body).

Otsikko-kenttä sisältää viestin käsittelyyn tarvittavan tiedon. Tällaista tietoa ovat reititys- ja välitystieto, tunnistus- ja käyttöoikeustiedot sekä kontekstitieto. Viesti-kenttä puolestaan sisältää lähetettävän viestin ja voi sisältää mitä tahansa tietoa, joka voidaan ilmaista XML-formaatissa.



Kuva 3.2. SOAP-viestin rakenne /TID01/

SOAP-viestin rakenteesta on esimerkki liitteessä 1, josta on nähtävillä viestin tarkka syntaksi. Viestin rakenne on hyvin selkeä, koska se on selkokielistä XML:ää. Selkeyden hintana on suorituskyvystä tinkiminen; kaikki viestit joudutaan ajamaan XML-tulkin kautta, mikä kuormittaa tulkkaavaa laitetta ja vähentää suorituskykyä /TAK05/. SOAP:n käsittely on hitaampaa kuin joidenkin jo olemassa olevien protokollien (RMI, CORBA) ja sen pullonkaulaksi on havaittu erityisesti viestien lukeminen ja parsiminen /TAK05/. Suorituskykyä käsitellään tarkemmin luvussa 3.3.

3.2.2 Palvelun kuvaaminen

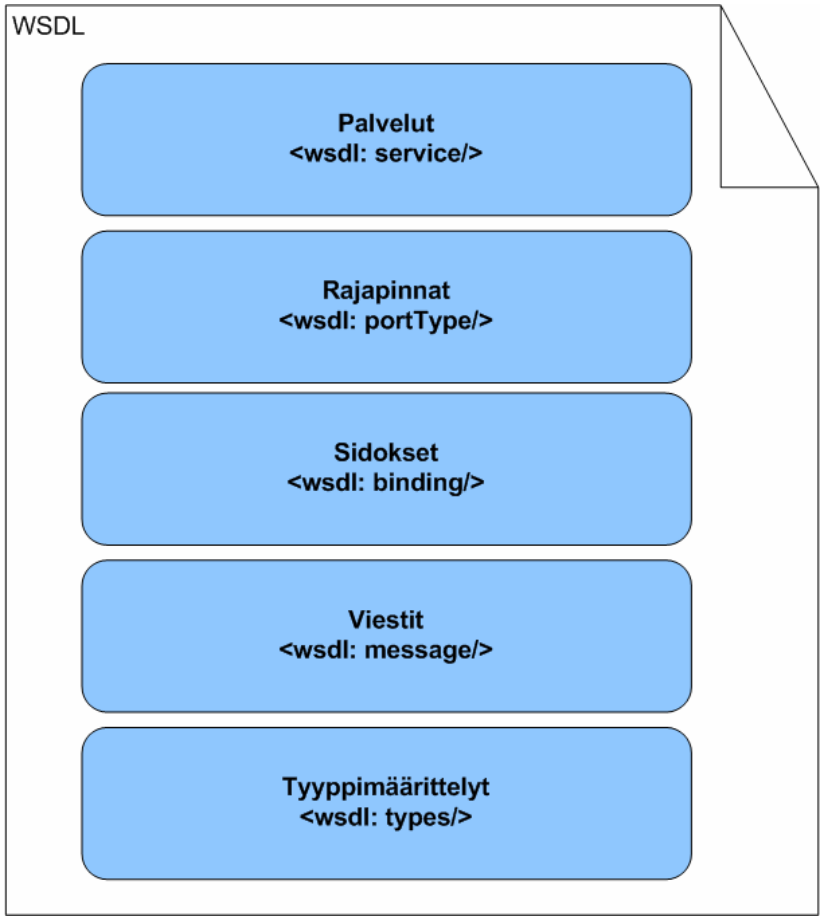
Yksi tärkeä ominaisuus Web Serviceissä on, että ne ovat itsensä kuvaavia. Tämä tarkoittaa, että ne paljastavat oman rakenteensa automaattisesti. Web Service kuvaa millaisia

funktioita se tarjoaa ja millainen protokolla on käytössä ilman, että asiakasohjelman täytyy tietää mitään palvelun toiminnasta etukäteen. SOAP ei hoida palvelun kuvaamista, vaan de facto-standardi tähän on WSDL.

WSDL kuvailee abstraktin rajapinnan, jonka kautta asiakas voi keskustella palvelun kanssa, sekä tarkemmat tiedot rajapinnan toteutuksesta. Käytännössä WSDL on kokoelma jatkuvasti tarkkenevia toisiinsa liittyviä määrityksiä. Se aloittaa kuvaamalla abstraktisti palvelun ja millaisia funktioita palvelu tarjoaa. Tämän jälkeen se jatkaa kuvaamalla konkreettisesti millaisia viestejä funktioissa liikutetaan ja millaisia ja minkä tyyppisiä muuttujia viesteissä on.

Käytännön esimerkki WSDL:stä on nähtävillä tämän dokumentin liitteestä 2, mutta yleisemmällä tasolla WSDL-määrityksen rakenne on esitetty kuvassa 3.3. Kuten kuvasta on nähtävillä, ensimmäisenä WSDL:ssä määritellään palvelu (eng. service). Palvelun määrittäminen pitää sisällään palvelun nimen ja kuvauksen sekä portit, joista se koostuu. Portit ovat osoitteita, joihin otetaan yhteys kun palvelua halutaan käyttää ja ne määrittelevät URL-osoitteeen (Uniform Resource Locator), johon otetaan yhteys palvelua käytettäessä.

Porteilla on sekä abstrakti määritelmä (portType), että konkreettinen määritelmä (binding) /TID01/. Abstrakti määritelmä määrittelee rajapinnan, joka koostuu operaatioista. Operaatiot puolestaan määrittelevät käytetyt viestit (eng. message). Konkreettinen määritelmä määrittelee käytetyt protokollat, kuten esimerkiksi SOAP. Tyyppimäärittelyillä määritellään tietotyypit ja ne kattavat vähintään kaikki XML-Schema-määrityksen mukaiset tietotyypit.



Kuva 3.3. WSDL-määrityksen rakenne /W3C03/

On hyvä huomata, että yleisesti ohjelmoijan ei tarvitse itse määritellä WSDL:ää, vaan tähän on olemassa valmiit työkalut. Esimerkiksi Microsoft Visual Studio ja myös monet avoimen lähdekoodin ratkaisut tuottavat WSDL-palvelunkuvauksen automaattisesti. Lähes kaikki kehitystyökalut sisältävät tuen WSDL:n tuottamiseen, joko suoraan tai laajennusten avulla. WSDL kannattaa tuottaa automaattisesti, sillä sen määritykset ovat hyvin pikkutarkkoja. Niiden koko kasvaa hyvin nopeasti suureksi, mitä enemmän toiminnallisuutta ja viestejä palveluun lisätään ja syntaksin on oltava tarkka. Käsin kirjoitetut määritykset ovat hyvin alttiita virheille, mutta koska määritykset johdetaan toteutetuista funktioista, niiden luominen on helppo automatisoida.

3.2.3 Palvelun etsiminen

Julkisen Web Servicen luominen edellyttää jonkinlaisia mekanismeja, joilla käyttäjät voivat löytää palvelun. Palvelun tarjoamisesta ei ole mitään hyötyä, jos sillä ei ole

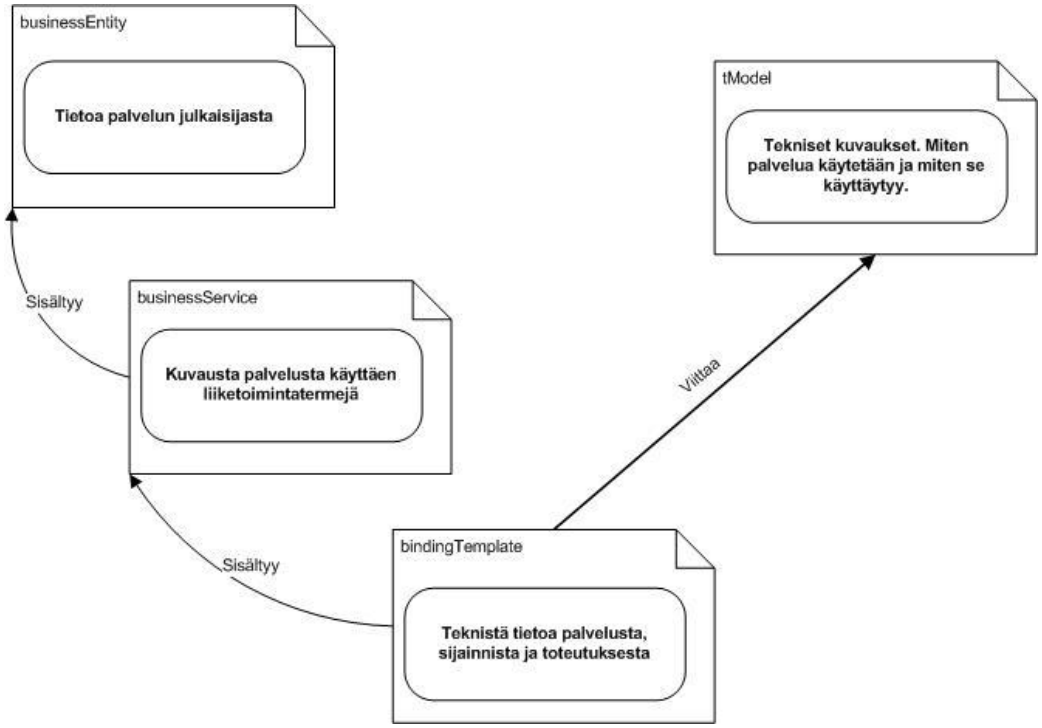
asiakkaita. Näiden löytämismekanismien (eng. discovery) tarjoamiseen on aloitettu Universal Discription, Discovery and Integration -projekti (UDDI). Projektin tarkoituksena on kasata rekisteri julkisista Web-palveluista ja niiden kuvauksista, jotta asiakkaat voivat etsiä vaivatta tarvitsemansa palvelut. Käytännön toteutukset UDDI:sta ovat hajautettuja tietokantoja, jotka pitävät sisällään meta-tietoa Web Serviceistä /BLA07/.

UDDI:lla on kaksi osaa: rekisteri Web Servicen metadatasta, mukaan lukien linkki sen WSDL-määritykseen, sekä rekisteri WSDL-porttimäärityksistä, joiden kautta palveluun voidaan luoda yhteys ja sitä voidaan käyttää. Rekisteri itsessään on määritelty hierarkiana yrityksistä, palveluista ja sidoksista, jotka on ilmaistu XML-kielellä /TID01/.

Käytännössä UDDI:n käyttäminen on kuin normaalin hakukoneen, kuten Googlen, käyttämistä. Palveluita voidaan hakea erilaisilla parametreilla ja esimerkiksi UDDI:n yritysrekisterin koostumuksen voi jakaa kolmeen osaan /SHA03/ :

- Valkoiset sivut: Sisältävät yrityksen yhteystiedot ja yrityksen tunnisteet. Sallii Web Servicen hakemisen yrityksen perusteella.
- Keltaiset sivut: Mahdollistavat yritysten listaukset niiden teollisuusalan mukaan.
- Vihreät sivut: Tekniset tiedot Web Servicen rajapinnasta.

Rekisterin sisältämä tieto voidaan jakaa viiteen tietorakenteeseen, kuten on esitetty kuvassa 3.4. BusinessEntity-rakenne sisältää tietoja palvelun julkaisijasta ja pitää sisällään eri tavoin luokiteltuja tietoja palvelusta (businessService- ja bindingTemplate). Se myös sisältää viitteen tModel-rakenteeseen, joka pitää sisällään tekniset määritykset palveluista /OASIS/. Nämä tyypit muodostavat koko UDDI rekisterin sisällön (UDDI v.2). Jokainen näistä XML-rakenteista sisältää useita kenttiä, jotka kuvaavat joko yritystietoja tai teknisiä tietoja.



Kuva 3.4. UDDI tietorakenteet /OASIS/

UDDI määritysten uusin versio kirjoitushetkellä on v.3. Uudet määritykset pyrkivät eriyttämään UDDI:a käsityksestä, että se olisi vain Web Serviceiden ”keltaiset sivut”. Se lisää tukea mm. digitaalisille allekirjoituksille, useille rekistereille sekä uuden rajapinnan palveluiden tilaamiselle. Toinen UDDI:n avulla saavutettava tärkeä ominaisuus on palveluiden tavoitettavuus: sen lisäksi, että voidaan vain etsiä olemassa olevia palveluita, voidaan niiden tila varmistaa reaaliajassa /BLA07/, eli voidaan varmistaa onko palvelu toiminnassa annetussa osoitteessa.

UDDI on oleellinen osa julkisia Web Serviceitä. Tämä on ymmärrettävää, kun palvelu halutaan tarjota mahdollisimman laajan käyttäjäjoukon saataville. Tämän diplomityön puitteissa toteutetulla Web Servicellä ei kuitenkaan ole tarvetta UDDI:lle. Rakennettaessa suljettua Web Serviceä, ei sen toiminnan yksityiskohtien julkaiseminen ole tarpeellista, tai edes suotavaa.

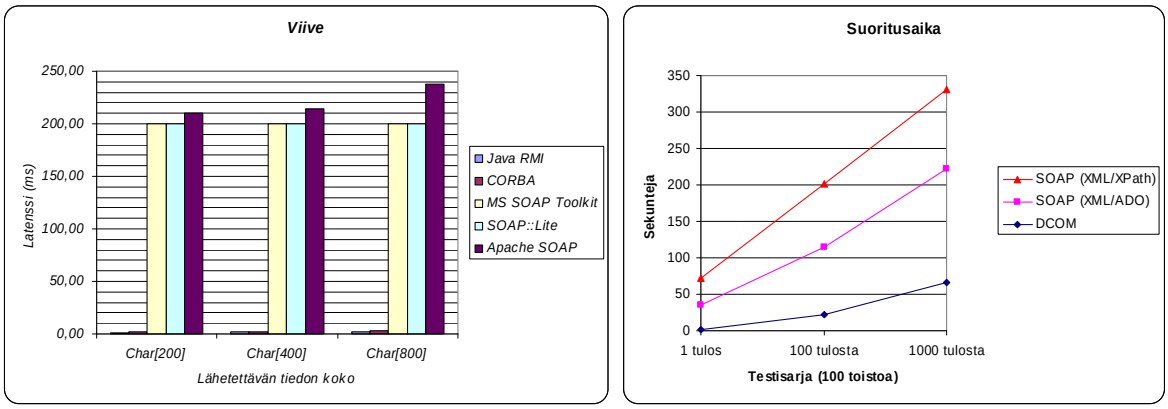
3.3 Suorituskyky

Web Servicen selkeydestä ja yhteensopivuudesta maksetaan pienempänä suorituskykynä.

Esimerkiksi hajautetussa laskennassa käytetään protokollia kuten CORBA (Common Object Request Broker Architecture), DCOM (Distributed Component Object Model) ja EJB:n (Enterprise Java Beans) tukemia protokollia, mutta Web Serviceiden suorituskyky ei riitä näihin käyttötarkoituksiin. Yksinkertaisempi RPC-protokolla (Remote Procedure Call), kuten aiemmin mainittu XML-RPC, tarjoaa parempaa suorituskykyä /WEI06/, mutta vastapainoksi käytetty protokolla täytyy sopia kommunikoivien ohjelmien ulkopuolella, vaatien yksityiskohtaista tuntemusta niiden toiminnasta. Tämä aiheuttaa ohjelmien kytkeytymistä. Koska avoimuus ja riippumattomuus ovat aina olleet Web Servicen lähtökohtia, tämä sotisi sen suunnitteluperiaatteita vastaan.

SOAP:n suorituskykyä on verrattu edellä mainittuihin protokolliin useissa tutkimuksissa ja tulokset ovat olleet hyvin samankaltaisia. SOAP on XML:n varaan rakentuva standardi ja XML-pohjainen siirtoprotokolla kärsii ison suorituskyky häviön, kun sitä vertaa binääritiedon siirtoon. Suorituskykyhäviön on pääasiassa katsottu syntyvän kahdesta seikasta:

- Järjestelmäkutsujen määrä viestin luomiseen. Tiedon kartoitus käytettävän kielen muuttujien ja XML-muuttujien välillä syö suorituskykyä.
- Vastaanotetun XML:n parsiminen ja lähetettävän XML:n formatointi



Kuva 3.5. SOAP-protokollan suorituskykyvertailua

Kuvassa 3.5 on esitelty kahden tutkimuksen tuloksia SOAP-protokollan suorituskyvystä verrattuna muihin ratkaisuihin. Tulokset ovat kahdesta eri artikkelista: viivettä eri

protokollien välillä testattiin artikkelissa ”Latency Performance of SOAP Implementations” /DAV02/ ja suoritukseen kuluvaa aikaa testattiin artikkelissa ”A Comparative Study of DCOM and SOAP” /ZHA02/. Suoritusaikatesteissä tehtiin 100 toiston sarjoja, joissa haettiin palvelun kautta tietokannasta tietoa vaihtelevilla tulosmäärillä. Yhden palautettavan tuloksen koko oli 40-60 tavua sekä XML-muotoilun tuoma lisä viestin kokoon. Viivetesteissä puolestaan vertailtiin SOAP:a RMI:n ja CORBA:n kanssa erilaisilla funktiokutsuilla. Tulokset kaikissa testeissä olivat hyvin samankaltaiset: latenssi SOAP-toteutuksissa on kautta linjan 20-kertainen. Suoritusajan vertailussa kävi ilmi, että vaikka erot eivät olleet yhtä suuria, häviää SOAP silti suoritusajassa. Haettaessa pieni määrä tuloksia, SOAP:n suorituskyky on jälleen noin 20-kertaa hitaampi, mutta suuremmilla tulosmäärillä erot tasaantuvat hieman. Palautettaessa 100 tai 1000 tulosta erot ovat jo pienemmät: tällöin SOAP on enää 3-4 kertaa hitaampi.

XML-parsimisen lisäksi SOAP-viestinnässä käytetty TCP-protokollaa on myös hidastava tekijä, sillä se saa aikaan suuren latenssin ja lisää ylimääräistä lähetettävää tietoa (eng. overhead) /PHA06/. Ongelmaa on yritetty ratkaista usealla tavalla ja sille on myös kysyntää erityisesti käytettäessä Web Serviceitä mobiiliympäristöissä. Tällöin protokollan tulisi olla mahdollisimman kevyt ja nopea. Yksi mahdollisuus suorituskyvyn parantamiseen on siirtotien valinta. SOAP määritys tarjoaa mahdollisuuden eri siirtoteille, esimerkiksi SOAP TCP:n, UDP:n tai SMTP:n yli, joiden käytöstä voi olla hyötyä erityisesti mobiilisovellusten kanssa /PHA06/. Tämä on yhä kokeellista, sillä ainoa tarkasti määritelty toteutus on SOAP HTTP:n yli. Suorituskyvyn parantamiseksi on myös tehty kokeellisia toteutuksia, kuten templaattien käyttö viestin muodostamisessa /WEI06/. Tällaisten optimointien on havaittu parantavat suorituskykyä, mutta ne ovat myös sijoituskohteesta riippuvaisia; ne eivät sovellu kaiken tyyppisiin palveluihin.

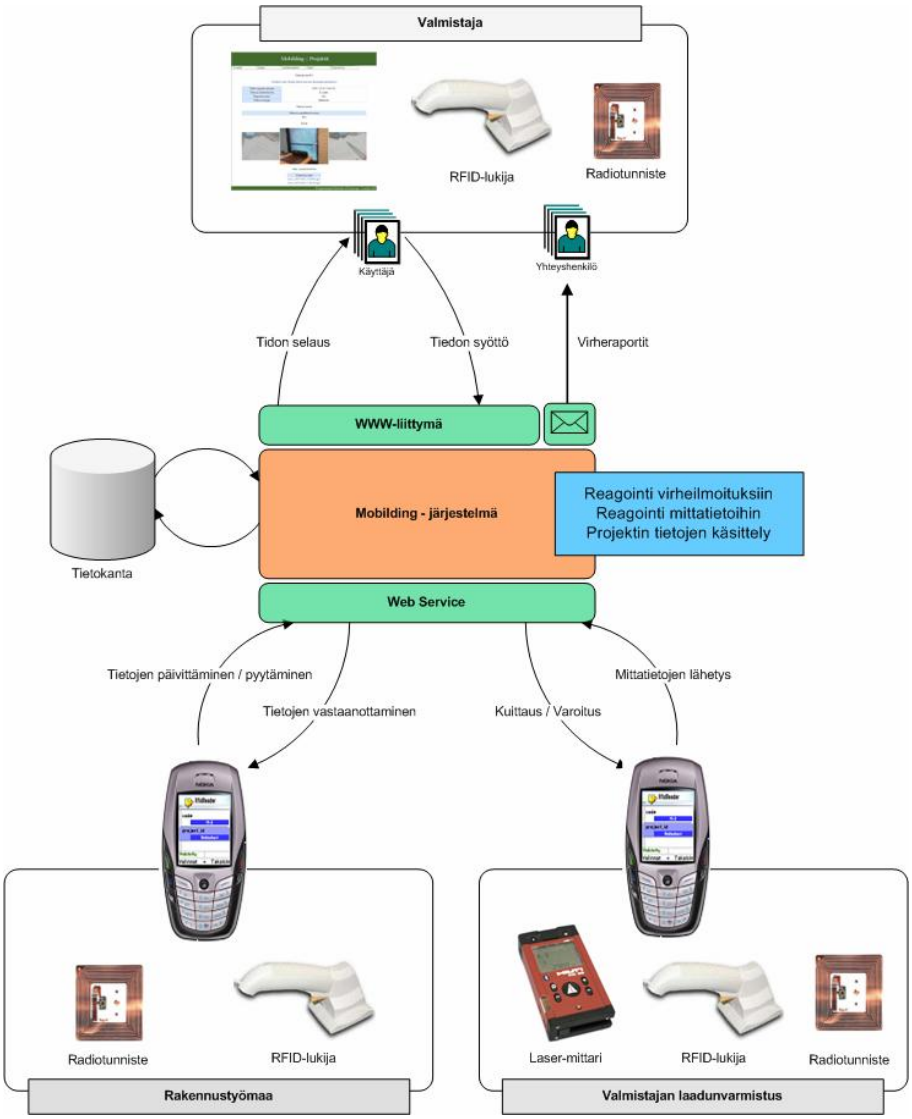
4. JÄRJESTELMÄN RAKENNE JA SUUNNITELUVALINNAT

Matkapuhelimen käyttö päätelaitteena on ollut Mobilding-hankkeessa yhtenä suunnittelulähtökohtana. Luvussa 2 tehdyssä nykytilanteen selvityksessä kävi ilmi, että suuri osa viestinnästä rakennustyömaan ja valmistajan välillä tapahtuu puhelimen välityksellä. Puhelin on koettu sekä helpoksi ja selkeäksi apuvälineeksi käyttää, että toiminnallisuudeltaan riittäväksi. Matkapuhelin on laite, jonka suurin osa ihmisistä omistaa; sitä käytetään päivittäin, joten se tarjoaa hyvin tutun liityntäpinnan uuteen järjestelmään. Helppokäyttöisyys on yksi tärkeimpiä kriteereitä, sillä tarkoituksena on tehostaa työntekoa ja käyttäjinä toimivat ihmiset, joilla ei ole välttämättä teknistä koulutusta.

Tässä diplomityössä toteutetaan laadunvarmistusjärjestelmä, joka kykenee ottamaan vastaan virheraportteja, havainnoimaan ongelmia aikataulutuksessa ja tiedottamaan poikkeamista ihmisiä, jotka ovat niistä vastuussa. Lähtökohtana tälle Mobilding-järjestelmälle on, että työntekijät voivat ottaa siihen yhteyden käyttäen matkapuhelinta. Matkapuhelin soveltuu hyvin kenttäolosuhteisiin, kuten rakennustyömaalle ja elementtitehtaan työalueille, mutta järjestelmän tarkempaan hallintaan tarvitaan myös käyttöliittymä, joka on toiminnaltaan rikkaampi. Diplomityö muodostaa kokonaisuuden Teemu Vilkon diplomityön "Mobiiliteknologia liikkuvan työntekijän apuna" /VIL07/ kanssa, jossa on toteutettu matkapuhelinohjelmisto Nokian S40-sarjan matkapuhelimille. Laadunvarmistusjärjestelmään toteutetaan Web Service-pohjainen rajapinta, joka mahdollistaa viestimisen mobiililaitteiden kanssa, logiikka poikkeustilanteiden hallintaan, jolla vastaanotettuun tietoon reagoidaan ja WWW-pohjainen käyttöliittymä, jolla järjestelmää voidaan hallita. Tässä luvussa käydään yksityiskohtaisesti läpi järjestelmän rakenne ja perustellaan tehdyt valinnat.

4.1 Laadunvarmistusjärjestelmä

Laadunvarmistusjärjestelmän käyttäjinä tulee olemaan sekä elementtien valmistaja, että rakennustyömaa, jonne elementit lähetetään. Molemmat käyttävät päätelaitteenaan matkapuhelinta, johon on kytketty RFID-lukija. Lukijan avulla elementit voidaan tunnistaa yksilöllisesti niihin upotettujen RFID-tunnisteiden perusteella. Tämän jälkeen Mobilding-järjestelmään voidaan lähettää virheraportteja, laser-mittarin avulla mitattuja mittatietoja tai kysyä tietoja elementistä. Yleiskuva järjestelmästä on nähtävissä kuvassa 4.1.



Kuva 4.1. Suunniteltu kaavio Mobilding-järjestelmästä

Mobilding-järjestelmä tallentaa tiedot tietokantaansa ja reagoi mittatietoihin ja

virheraportteihin lähettämällä niistä varoituksia sähköpostilla määritetyille yhdyshenkilöille. Järjestelmää hallitaan WWW-käyttöliittymän kautta, jonka toiminnallisuus on pääasiassa suunnattu elementtitehtaille. Sen kautta voidaan käsitellä virheraportteja, projekteja ja elementtien tietoja. Alussa käyttöliittymään toteutetaan tarvittu perustoiminnallisuus, mutta sitä tullaan kehittämään pilottihankkeissa saatujen kokemusten ja toiveiden mukaan.

4.1.1 Rajapinta matkapuhelimelle

Matkapuhelimella lähetettävän tiedon vastaanottamiseksi tarvitaan kommunikaation mahdollistava rajapinta palvelimen puolelle. Mobilding-palvelimella on jo olemassa rajapinta tiedon siirtämiseen Tekla Structures-rakennusmallin ja tietojärjestelmän välillä. Tämä rajapinta on kuitenkin toteutettu XML-RPC:llä, mikä ei ole jatkokehitystä ajatellen hyvä ratkaisu. Toteutus ei ole erityisen avoin ja se aiheuttaisi liikaa riippuvuutta päätelaitteen ja rajapinnan välillä: päätelaitteen on ennestään tiedettävä liian yksityiskohtaisesti rajapinnan toiminta.

Mahdollistaakseen avoimen toteutuksen ja erityisesti automaattiset palvelukuvaukset tarjotusta toiminnallisuudesta, rajapinta tulisi toteuttaa SOAP Web Service:n pohjalle. Näin päätelaitteiden, alustojen ja ohjelmointikielien vaihtaminen on helppoa: palvelukuvauksesta voidaan generoida helposti ohjelmarunko ja ei ole merkitystä millainen päätelaite palvelimen kanssa kommunikoi.

Rajapintaan toteutetaan funktiot puhtaasti tekstuaalisen tiedon lähettämiseen, kuten mm. virheraportit, mittatiedot ja tietokyselyt. Tämän lisäksi siinä täytyy olla mahdollisuus binääritiedon vastaanottamiseen ja lähettämiseen. Viesteissä tullaan siirtämään myös kuvia virheraportoinnin yhteydessä ja tulevaisuudessa mahdollisesti myös äänitiedostoja.

4.1.2 Palvelimen logiikka virheenhallinnassa

Mobilding-järjestelmään voidaan lähettää virheeseen johtavia viestejä kahdessa

tapauksessa: rakennustyömaalta lähetetään virheraportti tai tehtaan sisäisessä laadunvarmennuksessa lähetetään elementin mitatut mitat ja niiden havaitaan poikkeavan liikaa suunnitelluista mitoista. Kuvan 4.1 järjestelmän kuvauksessa on esitetty myös kommunikoinnissa ilmeneviä tapauksia. Palvelimen täytyy kyetä reagoimaan havaittuihin virheisiin ja tiedottamaan osapuolia poikkeustilanteista.

Kun palvelimelle lähetetään mittaustulokset, niitä tulee verrata hyväksyttäviin mittaustoleransseihin. Kaikki saadut mittaukset tulee tallentaa, jolloin tietoa ei hävitetä ja syötetty tieto on jatkuvasti saatavilla. Mikäli vastaanotetut mittatiedot ylittävät toleranssirajat, on mittaus tallennettava virheellisenä ja tunnistettava se mittavirheeksi. Rakennustyömaalta lähetettävistä virheviesteistä puolestaan voidaan heti tunnistaa, mikä virhe on kyseessä. Reagointi kummassakin tapauksessa hoidetaan samalla tavalla. Palvelin lähettää määritetylle sidoshenkilölle sähköpostia asiasta ja ilmoittaa ongelmatilanteesta. Kaikki raportit tallennetaan tietojärjestelmään ja niitä pitää pystyä käsittelemään myös WWW-liittymän kautta.

4.2 PHP:n laajennusvaihtoehdot rajapinnan toteutukseen

Järjestelmä toteutetaan käyttäen avoimeen lähdekoodiin perustuvia laajennuksia ja PHP-kieltä. Rajapinnan toteutukseen käytettävän SOAP-laajennuksen valinta on haastavaa, sillä PHP:lle toteutetut avoimet SOAP-ratkaisut ovat vielä kehitystilassa. PHP:lle on olemassa suljettuja ja kaupallisia ratkaisuja Web Servicen ja SOAP:n toteuttamiseen, mutta tässä diplomityössä haluttiin pitäytyä avoimissa ratkaisuissa.

Laajennusta valittaessa vertailtiin kolmea avointa toteutusta: PEAR::SOAP:a, NuSOAP:a, sekä PHP:n sisäänrakennettua SOAP-tukea. Kaikki laajennukset kykenevät perustason SOAP-viestintään, mutta niiden välillä on toiminnallisuudessa eroja. Varsinkin, jos toteutetussa palvelussa halutaan käyttää kehittyneempiä ominaisuuksia, kuten liitetiedostojen lähetystä, havaitaan kaikissa vaihtoehdoissa nopeasti vakavia puutteita. Seuraavissa kappaleissa esitellään kaikki kolme laajennusta ja keskustellaan niiden vahvuuksista ja heikkouksista.

4.2.1 PEAR::SOAP

PEAR (PHP Extension and Application Repository) on kirjasto avoimelle lähdekoodille, joka on tarkoitettu PHP:n käyttäjille /PEA07/. Se sisältää SOAP-laajennuksen, joka on ollut kehityksessä vuodesta 2002 saakka /SCH07/. Projekti on yhä beta-asteella ja uusin versio kirjoitushetkellä on 0.11.0. PEAR::SOAP on julkaistu PHP-lisenssin /PHPL3/ alla, joka sallii sen ilmaisen käytön riippumatta toteutuksen luonteesta.

PEAR::SOAP on ohjelmoitu PHP-kielellä, joten se on täysin suorituksenaikana tulkattavaa koodia. Se toimii myös vanhemman PHP version 4:n päällä, mutta vaatii tuekseen uusimmat PEAR-asennuspaketit. Se myös toimii ongelmitta PHP5:n omien SOAP funktioiden rinnalla, joten sen käyttöönotossa ei ole ongelmia. Suurimmat edut PEAR::SOAP:ssa ovat, että se tukee WSDL:n automaattista generointia, sekä liitetiedostoja. DIME- ja MIME-tyyppisten liitteiden lähettäminen SOAP-viestin yhteydessä ei aiheuta ongelmia asiakassovelluksen toteutuksessa, mitä kokeiltiin myös käytännössä.

Vaikka PEAR::SOAP on täysin avointa lähdekoodia, siitä on hankala löytää esimerkkejä toteutuksesta. Tämä tekee kehityksestä haastavaa. Projektin kotisivut tarjoavat lähinnä lähdekoodin tarkasteltavaksi, mutta eivät anna esimerkkejä, kuinka esimerkiksi toteuttaa palvelin- tai asiakassovellus. Vaikka yksinkertaisen sovelluksen tekeminen on helppoa, törmätään helposti ongelmiin käytettäessä mm. monimutkaisia tietotyyppejä ja liitetiedostoja.

Hyvistä puolistaan huolimatta PEAR::SOAP:ssa on omat ongelmansa. Vaikka se osaa generoida WSDL-kuvauksen, käytännön kokeilussa se osoittautui puutteelliseksi, erityisesti käytettäessä liitetiedostoja. Vaikka PEAR-laajennus tukee niiden lähettämistä ja vastaanottamista, se ei osaa generoida palvelun kuvausta näistä operaatioista oikein. Myös palvelun toteutuksessa havaittiin käyttäytymistä, joka todennäköisesti johtuu ohjelmiston beta-tilasta; jos funktioille mm. antaa liian monta parametria, palvelu ei osaa enää automaattisesti muotoilla palautettavia viestejä oikein. Yhteensopivuus saattaa myös

aiheuttaa ongelmia lähetettäessä monimutkaisia tietotyyppejä, varsinkin käytettäessä Microsoftin .NET-teknologialla toteutettuja SOAP-asiakasohjelmia. Osan näistä puutteista voi kiertää lisäämällä ylimääräisiä määrityksiä lähdekoodiin, mutta laajennuksessa on yhä paljon parantamisen varaa.

4.2.2 NuSOAP

NuSOAP on toinen laajennus SOAP-toiminnallisuuden lisäämiseen PHP-kieleen. Se on toteutettu PHP4:n aikana, jolloin PHP:ssä ei ollut sisäänrakennettua SOAP-toiminnallisuutta. NuSOAP on toteutettu kahtena PHP-luokkana, eli se on kirjoitettu puhtaalla PHP-koodilla, eikä sisällä ollenkaan käännettyä, natiivia koodia /NUS07/. Myös NuSOAP tukee WSDL:n generointia, sekä liitetiedostoja.

NuSOAP:n edellinen versio on julkaistu vuonna 2005 heinäkuussa /NUS07/. Sen saa toimimaan myös PHP5:n rinnalla, mutta tämä vaatii lähdekoodin muokkaamista. NuSOAP:n luokkanimet ovat ristiriidassa PHP5:n omien SOAP-luokkien kanssa, joten näiden nimiä täytyy muokata NuSOAP:n lähdekoodista. NuSOAP:n tukee liitetiedostoja, mutta tämä tapahtuu PEAR-laajennuksen avulla. Jotta liitetiedostoja voidaan käyttää SOAP-viestin yhteydessä, täytyy PEAR:n MIME-laajennus olla asennettu koneelle. NuSOAP:n keskustelualueet verrattuna PEAR::SOAP:n vastaaviin vaikuttavat huomattavasti aktiivisemmilta, joten avun ja esimerkkien saaminen on helpompaa.

4.2.3 PHP5 SOAP

PHP:n versio 5 toi mukanaan sisäänrakennetun tuen SOAP-pohjaisille Web Serviceille. Tässä luvussa esitellyistä kolmesta vaihtoehdosta PHP5:n sisäänrakennettu SOAP-tuki on ainoa, joka on käännetty (eng. compiled) osaksi PHP-kieltä. Koska muut vaihtoehdot ovat tulkittavaa koodia, voidaan olettaa että suorituskyvyn puolesta PHP5:n ratkaisu on tehokkain. PHP5 SOAP tukee sekä palvelimen, että asiakkaan määrittelyä WSDL-kuvauksen avulla. Käyttökokemusten perusteella PHP5:n ratkaisu on vakaa ja hyvin helppokäyttöinen. Koska se on myös sisäänrakennettu PHP5:een, on sen jatkokehitys

vakaammalla pohjalla kuin kahden edellä mainitun ratkaisun.

PHP5:n SOAP-tuella on kaksi suurta puutetta: WSDL-generointi ja liitetiedostot. Se ei tue näistä kumpaakaan /PHS07/. Jos rakennetaan vähänkään suurempaa Web Serviceä, WSDL:n automaattinen generointi on ehdoton vaatimus. PHP5:n tapauksessa epäilyttäväksi asian tekee, että PHP:n kehittäjä Zend on julkaissut tämän ominaisuuden Zend Studio-kehitysohjelmistossaan /ZEN07/. Kehitysohjelmisto on kuitenkin maksullinen sovellus ja vaikka generointia on selkeästi ajateltu, ominaisuutta ei ole lisätty PHP-tasolla lähdekoodiin. Asiakassovelluksen kanssa tällä ei luonnollisesti ole merkitystä, mutta palvelupäälle ominaisuus on kriittinen. Mikäli käyttäjät haluavat tehdä Web Servicen PHP5:n omalla SOAP-laajennuksella, vaihtoehtoina on määritellä vastaava palvelu joko NuSOAP:lla tai PEAR::SOAP:lla ja generoida WSDL-määritys näiden avulla tai käyttää ulkopuolista WSDL-luojaa (eng. generator).

Vaikka palvelunkuvaus saadaankin luotua automaattisesti, liitetiedostoja ei silti kyetä lähettämään. PHP5 ei tue MIME- tai DIME- liitteitä SOAP-viesteissä. Tätä ongelmaa voidaan kiertää esimerkiksi lähettämällä tiedoston sisältö base64-koodattuna normaalissa SOAP-tekstikentässä. Tämä kuitenkin paisuttaa SOAP-viestin kokoa suhteettomasti, sillä koko tiedoston sisältö joudutaan sisällyttämään viestin XML-rakenteeseen. Vaikka palvelun saisi näin toimimaan esimerkiksi WWW-selaimen kautta, se aiheuttaa ongelmia tietyissä asiakasympäristöissä, esimerkiksi matkapuhelimissa ja muissa mobiililaitteissa.

4.3 Rajapintalaajennuksen valinta

Kaikissa vaihtoehdoissa on omat ongelmansa, mutta Web Service päädyttiin toteuttamaan PEAR::SOAP-laajennuksella. Kolmesta tarjolla olleesta vaihtoehdosta PEAR::SOAP valittiin, koska se vaikutti sopivimmalta käyttökohteeseen. Kriittisimmät tarpeet palvelun luomiseen olivat tuki WSDL:n generoinnille ja liitetiedostoille. Toteuttamista kokeiltiin PHP5:n sisäänrakennetulla laajennuksella, mutta kehityksessä nousi tarve lähettää SOAP-viestien mukana liitetiedostoja. PHP5 ei yksinkertaisesti kyennyt tähän, jolloin jouduttiin tarkastelemaan muita vaihtoehtoja.

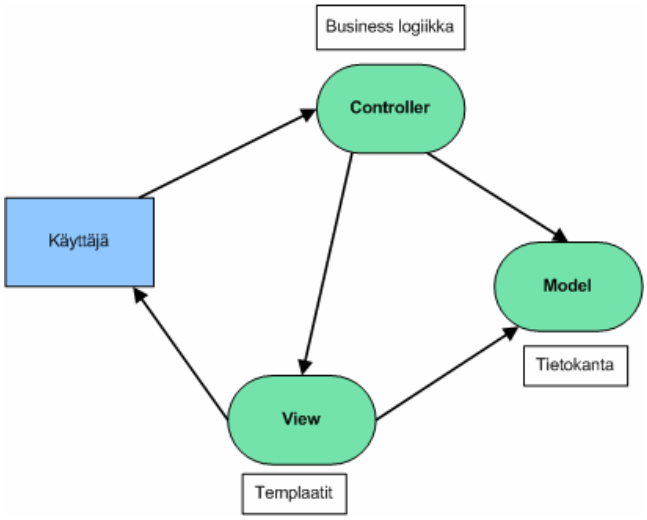
PEAR::SOAP ja NuSOAP tukevat molemmat teoriassa samoja ominaisuuksia. Lopullinen syy PEAR:SOAP:iin päätymisessä oli ylläpito: PEAR:n laajennusta kehitetään edelleen ja viimeisin päivitys kirjoitushetkellä oli julkaistu kesäkuun lopussa 2007 /SCH07/. Vastaavasti NuSOAP:lla ei ole ollut uutta versiojulkaisua yli kahteen vuoteen /NUS07/. Kommenteista voi päätellä, että kehitystä tapahtuu yhä postituslistoilla, mutta pelkästään se tosiasia, että NuSOAP ei toimi ilman lähdekoodin muokkausta PHP5:n rinnalla, aiheuttaa epäilyksiä sen soveltuvuudesta. Tämän lisäksi tiedostojen liittäminen SOAP-viesteihin vaatii PEAR:n MIME-laajennuksen asentamista toimiakseen, joten SOAP-kirjastot on joka tapauksessa asennettava, jos NuSOAP:a haluttaisiin käyttää. Mikäli PHP5:n omassa toteutuksessa tapahtuu kehitystä, on siirtyminen siihen varsin vaivatonta. Sinä aikana myös PEAR::SOAP todennäköisimmin tulee samaan korjauksia ja tukea.

4.4 Käyttöliittymän rakenne

Projektin tietojen hallintaan täytyy toteuttaa liittymäpinta, jonka avulla projektin ja elementtien tietoja voidaan hallita. Tämä toteutetaan WWW-liittymänä; se tarjoaa lähes universaalin ja helpon tavan päästä käsiksi tarvittuun tietoon. Käyttöliittymä ja tietojärjestelmä toimivat samanlaisen alustan päällä kuin rajapinnatkin. Palvelimelle toteutetaan sekä hallintasivustot, joilla käsitellään lähinnä käyttäjätietoja, että yritysten käyttöön tarkoitetut projekti-sivut. Kummatkin sivustot vaativat sisäänkirjautumisen ja käyttöoikeuksia valvotaan erityisesti valmistajien sivustoilla. Projekteissa voi olla useampia yrityksiä mukana ja on tärkeätä, että tietoa voivat tarkastella vain ne osapuolet, joilla on siihen oikeus. Yritykset voivat esimerkiksi tarkastella omia ja yhteistyökumppaniensa tietoja, mutta osa tiedoista, kuten esimerkiksi virheraportit, ovat arkaluonteisia, eikä kaikilla ole oikeuksia nähdä niitä.

Sivustojen toteutuksessa eri toimintatasot pidetään erillään ja suunnittelumetodina käytetään MVC-arkkitehtuuria (Model View Controller). MVC-arkkitehtuurin avulla kokonaisuus voidaan jakaa kolmeen loogiseen osaan: malliin (eng. model), joka sisältää tiedon, näkymään (eng. view), joka esittää tiedon käyttäjälle ja ohjaimeen (eng. controller),

joka on vastuussa sekä näkymistä, että mallista. Näin kokonaisuudesta saadaan selkeä järjestelmä, jota on helppo ylläpitää ja kehittää edelleen. Artikkelissa ”A Database and Web Application Based on MVC Architecture” /SEL06/ MVC-arkkitehtuurin hyödyiksi kuvaillaan sen kykyä antaa useita esitysmuotoja samasta tiedosta, rohkaisemista ohjelmakoodin uudelleenkäyttöön ja ohjelman jakamista osiin, antaen kehittäjälle mahdollisuuden keskittyä yhteen kokonaisuuteen kerrallaan. Artikkelin pohjalta MVC-arkkitehtuurin kolmijako esitetään kuvassa 4.2. Tärkein ominaisuus tässä jaossa on, että ulkoasu ja toimintalogiikka erotetaan toisistaan täysin.



Kuva 4.2. MVC-arkkitehtuuri /SEL06/

MVC-mallin mukaisesta käyttötilanteesta voi antaa myös esimerkin, kuinka asiakkaan pyynnöt käsitellään. Asiakas lähettää selaimellaan palvelupyynnön PHP-sivulle, jossa business-logiikka sijaitsee ja joka toimii ohjaimena (eng. controller). Se tulkitsee käyttäjältä saatavan syötteen ja sen perusteella käskee mallia (eng. model) ja näkymää (eng. view) muuttumaan tarpeen mukaan. Saapuvan syötteen perusteella ohjain esimerkiksi kutsuu tietokanta-objekteja ja hakee tai siirtää tiedot käytettävään tietokantaan. Tämän jälkeen ohjain järjestää tiedot määritettyyn muotoon ja muodostaa näkymän, joka esitetään käyttäjän selaimelle. Näkymä muodostetaan templaatin avulla, jossa ei ole enää suorituslogiikkaa, vaan sen avulla formatoidaan esitettävät tiedot haluttuun HTML-muotoon. Tätä prosessia tarkastellaan vielä tarkemmin dokumentin luvussa 6, jossa käsitellään laadunvarmistusjärjestelmän toteutusta.

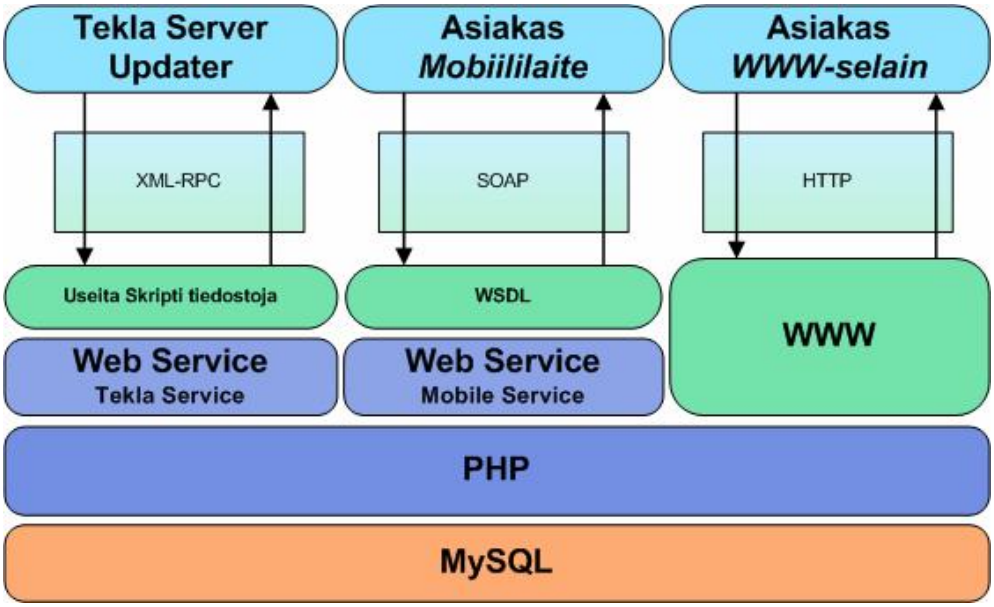
5. RAJAPINTOJEN TOTEUTUS

Tiedon syöttö Mobilding-järjestelmään tapahtuu mobiililaitteen, kuten esimerkiksi matkapuhelimen, avulla. Puhelimen kautta tapahtuva kommunikaatio tarvitsee toimiakseen rajapinnan ja tässä diplomityössä se toteutettiin Web Servicen muodossa. Rajapinta ei ota kantaa minkälainen päätelaite sen kanssa keskustelee, vaan sen voi olla käytännössä mikä tahansa yhteensopiva laite, jopa esimerkiksi normaali WWW-selain.

Tässä luvussa esitellään aluksi lyhyesti järjestelmän liittymäpinnat ja perehdytään sen jälkeen rajapinnan toteutukseen. Kappaleissa käydään läpi rajapinnan arkkitehtuuri, sekä toteutetut funktiot ja viestintä asiakasohjelmien välillä. Luvun 4 laajennusten vertailussa kävi jo ilmi, että rajapinnan toteutukseen käytetyissä laajennuksissa oli kaikissa omat puutteensa ja tämä tosiasia heijastui myös toteutuksessa. Se oli ajoittain haastavaa ja toteutukseen jäi muutamia rajoituksia, joita käsitellään tarkemmin luvun lopussa.

5.1 Mobilding-palvelimen liittymäpinnat

Järjestelmäkokonaisuus sisältää tällä hetkellä kolme erityyppistä liittymäpintaa tiedon siirtämiseen: kaksi eri teknologioilla toteutettua Web Service-liittymää ja selaimella käytettävän liittymän, jolla hallitaan järjestelmän tietoja. Yleiskuva liittymistä on nähtävillä kuvasta 5.1. Molemmat Web Servicet ovat tarkoitettu automaattiseen tiedon siirtoon päätelaitteiden ja järjestelmän välillä, mutta niiden asiakasohjelmat eroavat toisistaan. Näistä kahdesta Tekla Service-palvelu on toteutettu aikaisemmin ja se on luotu kommunikoimaan Tekla Server Updater-ohjelman kanssa. Tällä ohjelmalla voidaan siirtää Tekla Structures-rakennusmallinnusohjelmiston virtuaalimallin sisältö Mobilding-järjestelmään. Näin valmiista rakennussuunnitelmasta voidaan tuoda kaikki projektin elementit Mobilding-järjestelmän käyttöön. Tässä diplomityössä on toteutettu Web Serviceistä Mobile Service, jonka kautta suoritetaan tiedonsiirto projektissa käytettävien Nokian S40-sarjan matkapuhelinten kanssa.



Kuva 5.1. Mobilding-palvelimen liittymäpinnat

Kaikki liittymäpinnat toteutettiin PHP-kielellä ja käytetty versio oli 5.2.3. Vastaavasti kaikki liittymäpinnat keskustelevat saman tietokannan kanssa, joka on rakennettu MySQL version 5.0.45 päälle. Web Service-palvelut toteutettiin eri teknologioilla: vanhempi palvelu toteutettiin XML-RPC-tekniikalla, mutta sen puutteista johtuen matkapuhelinten kanssa kommunikoiva uudempi Web Service päädyttiin toteuttamaan SOAP-pohjaisena palveluna.

Kommunikaatioprotokollien lisäksi suurin ero Web Serviceiden toiminnassa on, että XML-RPC-Web Service:ssä jokainen funktio on pilkottu omaan tiedostoonsa, kun SOAP-pohjaisessa Web Servicessä kaikki toiminnallisuus on kootusti yhdessä tiedostossa. Tämä ominaisuus, lisättynä mahdollisuuteen julkaista WSDL-kuvaus palvelusta, tekee sen ylläpidosta yksinkertaisempaa ja asiakasohjelmien kehityksestä helpompaa.

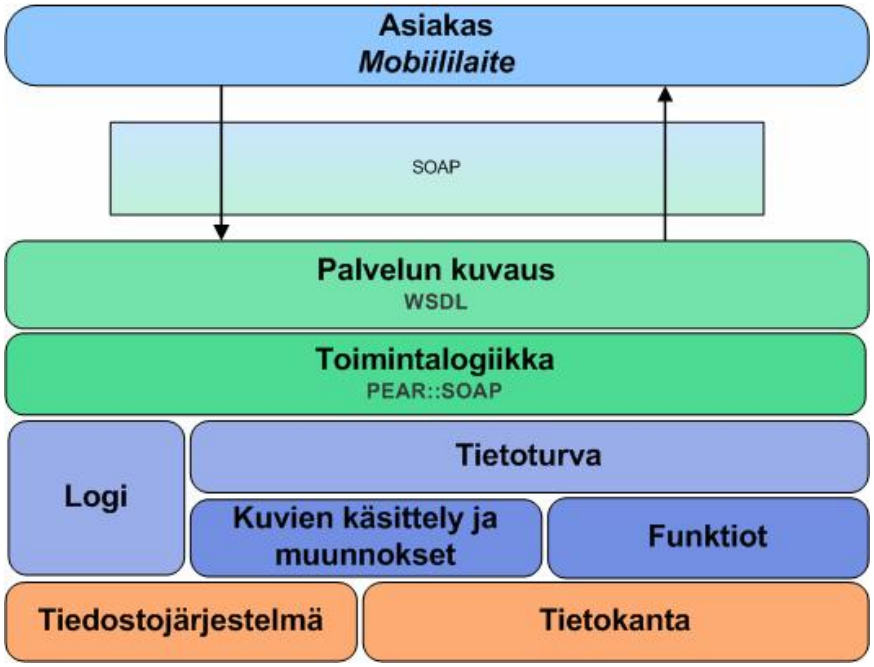
5.2 Mobile Service-rajapinta

Mobile Service-rajapinta tarjoaa liittymäpinnan Mobilding-tietojärjestelmään. Rajapinta on avoin ja sitä voidaan käyttää millä tahansa päätelaitteella, mutta tarjotut funktiot on tarkoitettu käytettäväksi matkapuhelimen käyttöliittymän kautta. Koko toiminnallisuus löytyy yhdestä URL-osoitteesta, johon matkapuhelin ottaa yhteyden.

Palvelu on avoimessa käytössä, mutta kaikkien funktioiden käyttö vaatii asiakkaan tunnistuksen. Asiakas ei voi käyttää niitä, ellei hänellä ole oikeuksia Mobilding-tietojärjestelmään. Seuraavissa kappaleissa käydään läpi Mobile Service:n arkkitehtuuri, rakenne ja siihen toteutetut funktiot tarkemmin.

5.2.1 Mobile Service-palvelun arkkitehtuuri

Mobile Service-palvelun arkkitehtuuri on yritetty pitää yksinkertaisena ja helposti ylläpidettävänä. Kaavio arkkitehtuurista on nähtävillä kuvasta 5.2. Rakenteen voi jakaa kolmeen osaan: asiakasohjelmalle näkyvä kuvaus palvelusta, palvelun sisällä suoritettavat funktiot ja tiedon hakeminen tai tallentaminen, joka tapahtuu joko tiedostojärjestelmään tai tietokantaan.



Kuva 5.2. Mobile Service-rajapinnan arkkitehtuuri

Palvelun kuvaus julkaistaan WSDL-määrityksenä, joka kuvailee yksityiskohtaisesti rakenteen, viestit ja käytettävät tietotyypit. Kuvaus generoidaan automaattisesti PEAR::SOAP:n avulla, jolloin palvelun laajentaminen on helppoa, eikä syntaksivirheitä

tarvitse pelätä. Tämä on oleellista, sillä Mobile Servicen määritykset ovat jo nykyisillä funktioilla pitkät: palvelun WSDL-kuvaus on 23 kb:n kokoinen XML-tiedosto, jossa on yli 300 riviä määrityksiä.

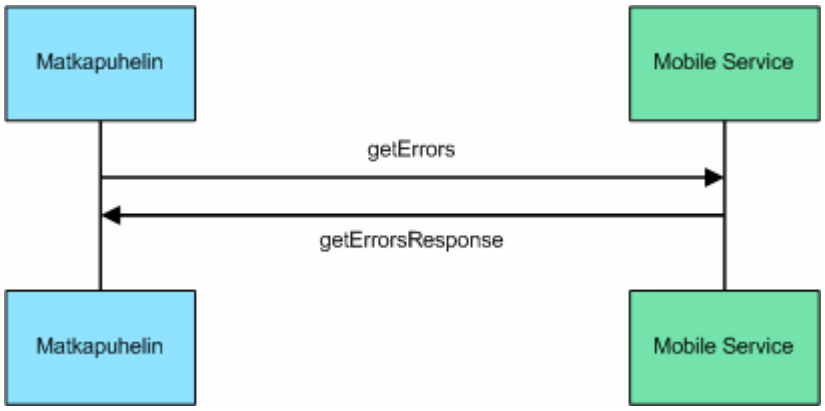
PEAR::SOAP kartoittaa määrityksen perusteella saapuvat kutsut palvelun sisäisen luokan funktioiksi. Tässä vaiheessa saapunut pyyntö prosessoidaan, mikä tarkoittaa käytännössä aluksi mahdollista pyynnön logiin kirjoittamista ja käyttöoikeuksien tarkistamista. Jos asiakkaalla on oikeudet pyyntöönsä, suoritetaan se. Kaikki käsiteltävät tiedot päätyvät tietokantaan tai lähtevät sieltä, poislukien karttatietojen pyytäminen palvelimelta ja liitetiedostojen lähettäminen, jotka käsitellään tarkemmin seuraavassa luvussa. Tällöin palvelimen tiedostojärjestelmästä joudutaan hakemaan kuvatietoa ja muuntamaan se XML-dokumenttiin kelpaavaan muotoon. Käytännössä tämä tarkoittaa, että lähetettävälle binääritiedolle suoritetaan base64-muunnos.

5.2.2 Lähetettävät viestit ja palvelun toiminnallisuus

Toteutettu palvelu tarjoaa funktioita tiedon hakemiseen ja yksittäisten elementtien tiedon päivittämiseen. Tunnistaminen tapahtuu aina elementtiin liitetyn tunnisteen perusteella, joka voi olla esimerkiksi RFID-tunniste tai viivakoodi, mikä on yhdistetty tietojärjestelmässä yksikäsitteiseen elementin tunnukseen. Mobiililaite lähettää aina tunnistekoodin viestin mukana. Palvelu yhdistää RFID-tunnuksen tietojärjestelmässä sitä vastaavaan elementtiin tai palauttaa virheen, jos tämä ei onnistu. Jokaisen viestin mukana kulkevat myös käyttäjätunnus ja salasana, jolla asiakas tunnistetaan ja varmistetaan, että hänellä on oikeus pyytämänsä toiminnallisuuden käyttöön. Palvelussa lähetettävien viestien päätyypeistä on esitelty MSC-kaaviot tässä luvussa ja lopuista viesteistä on nähtävillä vastaavat kaaviot liitteessä 3. Viestien tarkemmasta sisällöstä on tarkasteltavissa WSDL-kuvaus diplomityön liitteessä 2.

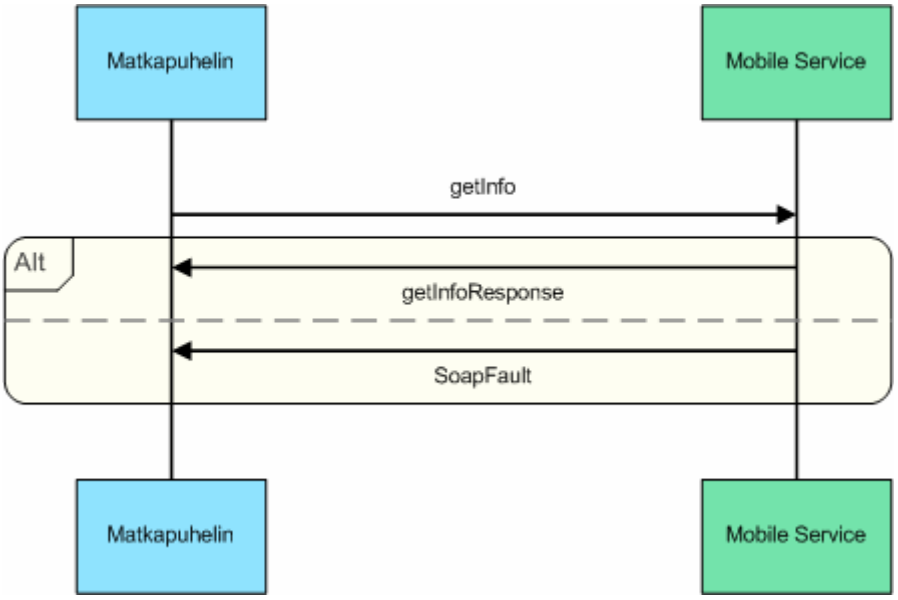
Viestinnän tarkasteleminen voidaan aloittaa virheraportoinnista. Elementille voidaan merkitä järjestelmään virheitä, joita siinä on havaittu. Tällainen virhe voi olla esimerkiksi elementeissä havaittu halkeama. Virhetyyppejä käsitellään tarkemmin vielä luvussa 6, mutta virheiden hakemiseen käytettyä getErrors-viestiä, joka on nähtävillä kuvasta 5.3,

voidaan käyttää esimerkkinä yleisimmästä viestimuodosta. Pääosa palvelun viesteistä on tämän rakenteen mukaisia: matkapuhelin lähettää viestin kysyäkseen tai päivittääkseen tietoa ja saa vastauksekseen viestin, joka sisältää virhe/onnistumiskoodin, sekä funktioon liittyvää tietoa. Esimerkkitapauksessa paluuviesti sisältää onnistumiskoodin, sekä lista-rakenteeseen (eng. Array) tallennetut elementille rekisteröidyt virheet. Yleisesti ottaen funktiot palauttavat vastausviestissä tiedon operaation onnistumisesta boolean-arvona ja mahdollisen virheviestin, joka kuvailee tarkemmin, miksi operaatio ei onnistunut.



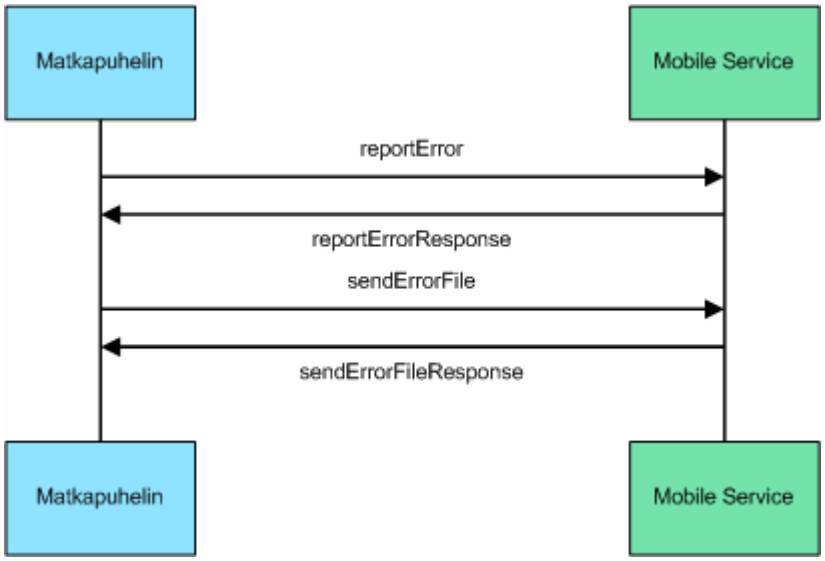
Kuva 5.3. getErrors-viestin MSC-kaavio

Poikkeuksen edellä mainittuun tapaukseen muodostavat getStrorages- ja getInfo-viestit. GetStorages-viesti hakee listauksen elementtien varastopaikoista elementtityypin perusteella ja getInfo-viesti puolestaan hakee järjestelmästä elementin tiedot, kuten dimensiot ja mahdolliset virheet. Näille viesteille ei ole paluuviestissä määritelty kenttää onnistumiselle, kuten on nähtävillä getInfo-viestin tapauksessa kuvasta 5.4. Jos operaatio epäonnistuu, se lähettää paluuviestissä standardin SOAP_Fault-viestin, joka on määritelty SOAP-protokollan virallisessa määrityksessä. Viesti sisältää virhekuvauksen tapahtuneesta ja sen ymmärtäminen ei ole ongelma millekään asiakkaalle, joka tottelee SOAP-määrityksiä. SOAP_Fault-viesti palautetaan asiakkaalle myös aina, jos sen lähettämä viesti ei ole palvelun WSDL-määrityksen mukainen.



Kuva 5.4. getInfo-viestin MSC-kaavio

Myös virheen raportointi on poikkeustapaus, jossa joudutaan lähettämään useampi viesti. Tästä viestinnästä on esimerkki kuvassa 5.5. Koska virheraporttiin voidaan liittää tiedostoja, joiden tyyppi ja määrä ovat vapaasti määriteltäviä, lähetetään ne itse virheraporttiviestin ulkopuolella. Ensimmäisenä lähetettävä reportError-viestille palautetaan onnistumisen yhteydessä virheraportin tunnus, joka puolestaan yhdistetään sendErrorFile-viestiin. Näin yhdelle raportille voidaan liittää rajaton määrä liitetiedostoja.



Kuva 5.5. Virheenraportoinnin MSC-kaavio

Seuraavissa taulukoissa on esitelty kaikki rajapinnan funktiot ja niiden toiminnallisuus lyhyesti. Taulukossa 5.1 on nähtävillä elementtien tiedon päivittämiseen tarkoitetut viestit. Näillä funktioilla elementteihin voidaan liitää uutta tietoa tai vanhoja tietoja voidaan päivittää. Funktiot muokkaavat vain yksittäisen tunnisteen määrittämän elementin tietoa, eli useiden elementtien tietoa ei voi muuttaa samaan aikaan.

Viestin nimi	Kuvaus toiminnasta
sendLocation	Lisää elementille senhetkiset GPS-koordinaatit.
sendState	Päivittää elementin tilan. Esimerkiksi kuittaa elementin valmistuneeksi tai asennetuksi.
reportError	Ilmoittaa elementissä havaitusta virheestä ja luo uuden sanallisen virheraportin.
sendErrorFile	Lisää tiedoston (kuva/ääni) luotuun virheraporttiin.
updateErrorState	Päivittää virheen tilan. Esimerkiksi kuittaa ilmoitetun virheen korjatuksi.
sendMeasurements	Lähettää elementin lasermittarilla mitatut dimensiot. Jos lähetetyt dimensiot eivät ole toleranssirajojen sisällä, paluuviestissä varoitetaan asiasta.

Taulukko 5.1. Tiedon päivittämiseen tarkoitetut viestit

Tietojen hakemiseen Mobilding-järjestelmästä on tarkoitettu taulukossa 5.2 esitellyt viestit. Näillä funktioilla voi tiedustella yksityiskohtaisempia tietoja joko elementeistä tai tietyn tyyppisistä elementeistä. Funktiot palauttavat monimutkaisempia vastauksia kuin päivitykset: pyydettävät tiedot palautetaan listoina ja esimerkiksi getMapLocation-funktion tapauksessa vastaus sisältää kuvatiedoston.

Viestin nimi	Kuvaus toiminnasta
getInfo	Palauttaa tiedot elementistä: mm. dimensiot, mittaukset, tilan sekä mahdollisen listan elementissä ilmenneistä virheistä.
getStorages	Palauttaa listan sijainneista, mistä kysyttyä elementtityyppiä löytyy. Lista sisältää sanallisen kuvauksen ja GPS-koordinaatit sijainneista.
getErrors	Palauttaa, mitä virheitä elementille on raportoitu.
getMapLocation	Palauttaa karttakuvan, missä kysyttyä elementtityyppiä on varastoitu.

Taulukko 5.2. Tiedon hakemiseen tarkoitetut viestit

5.3 Ongelmat Web Service rajapinnan toteutuksessa

Suurimmat ongelmat Web Service-rajapinnan toteutuksessa olivat käytetyn laajennuksen

dokumentaation puute ja sen toteutuksessa ilmenneet puutteet. Yksinkertaisen Web Servicen toteutus PHP:llä on helppoa. Riippumatta mitä SOAP-laajennusta käyttää, toimintalogiikka ja viestit generoidaan aina automaattisesti. Tästä syystä monimutkainen logiikka ja itse toteutetut funktiot voidaan peittää laajennuksen alle ja toiminnallisuus voidaan siirtää sellaisenaan, jos halutaan käyttää jotain muuta PHP:n laajennusta.

Ongelmat ilmenevät, jos halutaan luoda Web Service, jossa on monimutkaisempia ominaisuuksia. Tällaisilla ominaisuuksilla tarkoitetaan esimerkiksi tiedostojen liittämistä tai monimutkaisten tietorakenteiden käyttämistä viesteissä. Seuraavat kappaleet käsittelevät kahta havaittua ongelmaa: liitetiedostojen lähettämistä, sekä yhteensopivuutta viestinnässä.

5.3.1 SOAP liitetiedostot

Käytetty laajennus, PEAR::SOAP 0.11.0, kykenee lähettämään tiedostoja MIME- tai DIME-liitteinä SOAP-viesteissä. Jos laajennusta käyttäen tehtäisiin asiakassovellus Web Serviceen, ongelmaa tuskin tulisi. Kun PEAR::SOAP:lla kuitenkin toteutetaan palvelinsovellus, tilanne monimutkaistuu. Ongelmana ei niinkään ole itse viestin rakenne. Palvelinsovelluksesta pystytään palauttamaan oikeassa formaatissa oleva viesti, jossa XML-pohjainen selkokielinen viesti ja tiedoston binääri-tieto on eroteltu selkeästi toisistaan. Tämä viesti pystytään lukemaan esimerkiksi PEAR::SOAP:lla toteutetulla asiakassovelluksella ilman ongelmia. Ongelmaksi muodostuu palvelukuvauksen generoiminen. Laajennus ei pysty tuottamaan oikeanmuotoista kuvausta funktiosta, joka esittäisi palautettavaa viestiä standardin mukaisesti. Täten sen vastaanottaminen asiakaspäässä muuttuu mahdottomaksi, jos se on tehty palvelunkuvauksen perusteella. Luonnollisesti asiakasohjelmistoa voidaan muokata lähdekooditasolla niin, että se osaa tulkita tiedot viestistä, mutta kommunikointi ei tällöin enää tapahdu automaattisesti kuvauksen perusteella ja siten sotii koko Web Servicen suunnitteluajatusta vastaan. Tämä myös monimutkaistaa asiakassovellusten toteuttamista tarpeettomasti.

Ongelma liitetiedoston lähetyksessä kierrettiin lähettämällä tiedoston sisältö suoraan

viestin XML-rakenteen string-kentässä. Tiedoston sisältö luetaan palvelimella ja binääritiedolle suoritetaan base64-muunnos. Näin tieto saadaan muunnettua ASCII-muotoon, joka ei riko XML-viestin rakennetta. Viestin vastaanottaja puolestaan suorittaa käänteisen muunnoksen vastaanotettuaan viestin. Koska base64 on hyvin yleinen muunnosalgoritmi, jota käytetään mm. sähköposteissa ja MIME-käännöksissä, on turvallista olettaa, että tästä suoriutuvan asiakasohjelman toteuttaminen on mahdollista. Näin palvelimelta palautettava viesti saadaan noudattamaan palvelunkuvausta ja palvelu säilyy avoimena.

5.3.2 Yhteensopivuus eri laajennusten kanssa

Toinen suurempi ongelma mihin törmättiin, on yleinen yhteensopivuus viestinnässä. Web Serviceiden tulisi jo suunnittelulähtökohdasta katseltuna olla mahdollisimman yhteensopivia; se on kantava ajatus, jonka varaan ne on rakennettu. Kuten edellisessä luvussa mainittiin, eri SOAP-laajennusten viestien muotoilussa on eroja. Tässä diplomityössä tehtyä toteutusta testattiin eri laajennuksin toteutetuilla asiakasohjelmilla (PHP5 ja MS Visual Studio). Tulokset olivat vaihtelevia, eikä kommunikointi aina onnistunut. Ongelmatilanteissa asiakasohjelmia voi muokata niin, että ne parsivat tiedon lähetetyistä viesteistä. Tämän tulisi kuitenkin tapahtua automaattisesti suoraan palvelun WSDL-määrityksen perusteella.

Yksi syy yhteensopivuusongelmiin on, ettei lähetetty viesti vastaa täysin laajennuksen generoimaa WSDL-kuvausta. Yksinkertaisilla tietotyypeillä ei yhteensopivuudessa ollut minkäänlaisia ongelmia, mutta käytettäessä listoja tietotyyppinä, yhteensopivuus ei ollut enää taattua. PEAR::SOAP:lle täytyy muutamassa tilanteessa antaa käsin lisämäärityksiä, jotta se osaa julkaista oikean muotoisen WSDL-kuvauksen listoista. Lisämääritysten käyttö on kuitenkin ilman lähdekoodin muokkaamista rajoitettua, eikä esimerkiksi auta liitetiedostojen tapauksessa.

PEAR::SOAP:lla oli ongelmia mm. kartoittaa sisäkkäisten listojen (eng. nested array) tietotyypit oikein palvelunkuvauksen mukaisesti. Tämän lisäksi palautettavan viestin

rakenne riippui myös muuttujien määrästä: jos palautettava viesti sisälsi vain yhden lista-tyyppisen (eng. array) muuttujan, viestin ja listan rakenne oli erilainen kuin lähetettäessä esimerkiksi kaksi muuttujaa: yksinkertainen tietotyyppi ja lista. Tätä ei tulisi tapahtua, vaan kummassakin tapauksessa listan tulisi olla nimetty ja muotoiltu samoin tavoin.

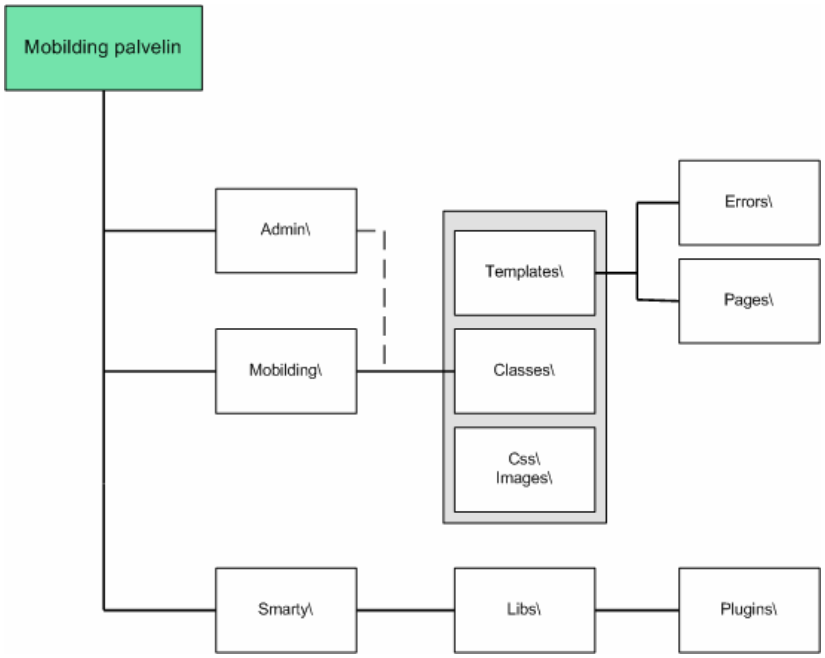
Käytettäessä Microsoft Visual Studion laajennusta ilmeni myös ongelmia listojen luvussa. Tämä näytti olevan yleistä myös muilla avoimilla PHP-laajennuksilla. PEAR::SOAP:ssa tilanne on kuitenkin korjattavissa. Palvelun kuvaukseen täytyy käsin määrittää listojen tapauksessa lähetettävien listojen alkioiden määrä. Microsoftin SOAP-toteutus tarvitsee tämän tiedon parsiakseen viestin oikein ja jos se on määritelty, pitäisi laajennusten toimia yhteensopivasti

6. LAADUNVARMISTUSJÄRJESTELMÄN TOTEUTUS

Laadunvalvontajärjestelmän ytimenä on logiikka, miten se reagoi saatuun tietoon. Edellisessä luvussa syvennyttiin tarkastelemaan, kuinka tietoa siirrettiin Mobilding-järjestelmän ja mobiililaitteen välillä. Tässä luvussa käydään läpi mitä tiedolle tehdään, kun se on järjestelmän sisällä ja kuinka toteutettu laadunvarmistusjärjestelmä toimii. Kappaleissa esitellään kuinka poikkeamien havaitseminen ja niistä ilmoittaminen toimii käytännössä ja pohditaan toteutuksessa ilmenneitä haasteita, kuten järjestelmän joustavuutta reaalimaailman tilanteisiin. Toimintalogiikan lisäksi kappaleissa käsitellään myös järjestelmän hallintaa. Hallinta on oleellinen osa kokonaisuutta, sillä ollakseen hyödyllinen, järjestelmän tulee olla helppokäyttöinen ja esittää tieto ymmärrettävässä ja havainnollisessa muodossa. Graafisella käyttöliittymällä on tässä osassa tärkeä rooli.

6.1 Vaatimukset ja rakenne

Järjestelmä on toteutettu käyttäen samoja teknologioita kuin aiemmin esitelty Mobile Service-rajapintakin. Järjestelmä toimii Apache-WWW-palvelimen varassa ja kielenä toteutuksessa on käytetty PHP:tä. Palvelimen rakenne on jaettu loogisesti omiin osiinsa, kuten on nähtävillä kuvassa 6.1. Sivujen toiminta on rakennettu Smarty Template Enginen päälle (versio 2.6.8), jonka avulla ulkoasu ja itse ohjelmakoodi erotetaan toisistaan. Käytännössä fyysisen rakenteen voi jakaa kolmeen osaan: hallinta-sivusto (Admin), yritysten sivustot (Mobilding), sekä Smarty ja sen lisäosat. Kuvassa katkoviiva ilmaisee, että rakenteeltaan hallinta- ja Mobilding-hakemistot ovat samanlaisia ja toimivat samojen periaatteiden mukaan.



Kuva 6.1. Käyttöliittymän rakenne palvelimella

Smartyn avulla voidaan käsitellä templaatteja. Ne ovat käytännössä WWW-sivujen runkoja, jotka eivät sisällä PHP-kieltä ollenkaan /SMART/. Ne kuvaavat sivun rakenteet (HTML, CSS, JavaScript) ja sisältävät viitteen dynaamiselle tiedolle, jonka Smarty täyttää, kun sivu lähetetään asiakkaan selaimeen. Käytännössä templaatti sisältää esimerkiksi kaikki tyylilliset määritykset HTML-rakenteelle, mutta ei rakenteen sisältöä tai PHP-funktioita, joilla se täytettäisiin. Kun asiakas pyytää sivua, toteutettu ohjelmakoodi hakee tiedon tai tekee halutut toimenpiteet ja antaa tiedon Smarty:lle, joka täyttää tiedon templaatin määrittelemään rakenteeseen ja palvelee sen asiakkaalle. Näin PHP-ohjelmakoodi pysyy tiukasti omissa tiedostoissaan ja esitykseen käytettävät kielet pysyvät erillään templaatissa.

Toimintalogiikan ja ulkoasun erottaminen helpottaa tiedon ja tiedostojen loogista järjestelyä (osat pysyvät erillään, templaatit ja tiedostot on helpompi järjestellä). Toinen saatava hyöty on, että templaattien avulla voidaan luoda erilaisia näkymiä. Sivustolla on eri käyttäjäryhmiä (eri yrityksiä), joilla on erilaisia tarpeita. Eri yritykset joutuvat syöttämään elementtien tiedot eri tavalla järjestelmään, johtuen eroista yritysten kirjanpidossa ja rakennussuunnitelmissa sekä pilottihankkeiden vaatimuksista. Koska sivut ovat templaatteina, voidaan suorituslogiikassa tarkastaa, mihin yritykseen käyttäjä kuuluu ja

tarjota sen perusteella erilaisia näkymiä samoihin toimintoihin templaattien avulla. Näin voidaan käyttää samaa business-logiikkaa, joka saattaa olla hyvin samankaltainen eri yrityksille tarjottavissa operaatioissa ja vain vaihtaa sivun ulkoasu palvellessa sitä asiakkaalle.

Smartyyn on myös saatavilla laajennuksia, joilla voidaan automatisoida monia mekaanisia toimenpiteitä. Tässä työssä käytetään Smarty Validate-laajennusta /SMAVA/, jolla voidaan tarkistaa WWW-sivuilta annettavien syötteiden oikeellisuus; ovatko ne vaaditussa muodossa vai eivät. Smarty Validaten avulla tästä tarkistamisesta saadaan hallitumpaa ja yksinkertaisempaa. Molempien ohjelmien lisenssit sallivat niiden käytön ilmaiseksi sekä suljetuissa, että avoimen lähdekoodin toteutuksissa, joten niiden käyttö tämän diplomityön tapauksessa on turvallista. Smarty Template Engine on lisenssoitu GNU Lesser General Public-lisenssin /LGPL/ alle, joka toisin kuin GPL-lisenssi, ei vaadi lähdekoodin jakamista. Smarty Validate on lisenssoitu PHP-lisenssin alle /PHPL3/.

6.2 Käyttäjien tunnistus

Toteutettu palvelu sisältää luottamuksellista tietoa sekä useita käyttäjäryhmiä. Tästä johtuen palveluun täytyi toteuttaa käyttäjien tunnistus (eng. authentication); sivujen ja järjestelmän käyttöoikeuksia täytyi valvoa. Tätä varten luotiin oma PHP-luokka. Jokaisella sivupyynnöllä tarkastetaan käyttäjän oikeudet ja varmistetaan, onko hänellä oikeus pyytämäänsä tietoon. Tunnistusta myös käytetään määrittämään millainen näkymä käyttäjälle esitetään, riippuen mihin käyttäjäryhmään hän kuuluu.

Tunnistus perustuu sessioiden käyttöön. Kun käyttäjä kirjataan sisään järjestelmään, tunnistus-luokka hakee eri tietokannan tauluista usein tarvittavat kriittiset tiedot käyttäjästä ja tallentaa ne sessio-tauluun tietokannassa, luoden samalla uniikin sessiotunnuksen. Tällaisia tietoja ovat käyttäjän nimi, sidosyritys ja käyttöoikeudet eri projekteihin. Kun käyttöoikeudet ja usein tarvittavat tiedot tämän jälkeen tarkistetaan jokaisella sivun latauksella, vaatii tarkastus vain yhden tietokantakutsun yhdestä tietokannan taulusta ja järjestelmän suorituskyky ei kärsi tarkastuksista.

Käyttäjät yhdistetään omiin sessioihinsa tallentamalla käyttäjän selaimeen cookie-tunniste, joka sisältää sessiotunnuksen. Tallennettava sessiotunnus ei sisällä itsessään mitään selkokielistä tietoa, jonka perusteella käyttäjä voitaisiin tunnistaa palvelun ulkopuolella, vaan se on tietyin parametrein laskettu yksilöllinen md5-hash summa. Tämä luodaan uudestaan aina jokaisen sisäänkirjautumisen yhteydessä.

6.3 Laadunvarmistus

Laatu merkitsee eri osapuolille eri asioita: elementtitehtaalle se on pitkälti kustannusten minimointia ja virheiden välttämistä. Rakennustyömaalla se myös merkitsee työajan optimointia; vaikka elementtitehdas korvaisi viallisen elementin, pahimmassa tapauksessa rakentaminen pysähtyy ja työaikaa hukataan. Toteutetussa järjestelmässä laadunvarmistusta lähdettiin automatisoimaan virheraportoinnin ja elementtien tilatietojen seuraamisen avulla.

Virheitä voidaan aiheuttaa järjestelmässä kahdella tavalla: reagointina saatuihin viesteihin tai säännöllisesti tehtyinä tarkastuksina elementin tilatietoihin. Tilatietojen tarkistaminen tarkoittaa sen nykyisen tilan vertaamista suunniteltuihin tiloihin. Elementillä voi olla esimerkiksi suunniteltu asennuspäivä rakennustyömaalla ja jos elementin nykyinen tila ei ajanhetkellä vastaa suunniteltua, aiheuttaa se virhetilanteen.

Virheistä ilmoitetaan vastuuhenkilöille sähköpostiviestein ja ilmoittajalle paluuviestinä matkapuhelimeen. Seuraavat kappaleet käsittelevät yksityiskohtaisemmin laadunvarmistusprosessin eri osia Mobilding järjestelmässä. Toteutuksessa tulleita haasteita puolestaan käsitellään myöhemmin luvussa 6.4.

6.3.1 Virheiden kirjaaminen järjestelmään

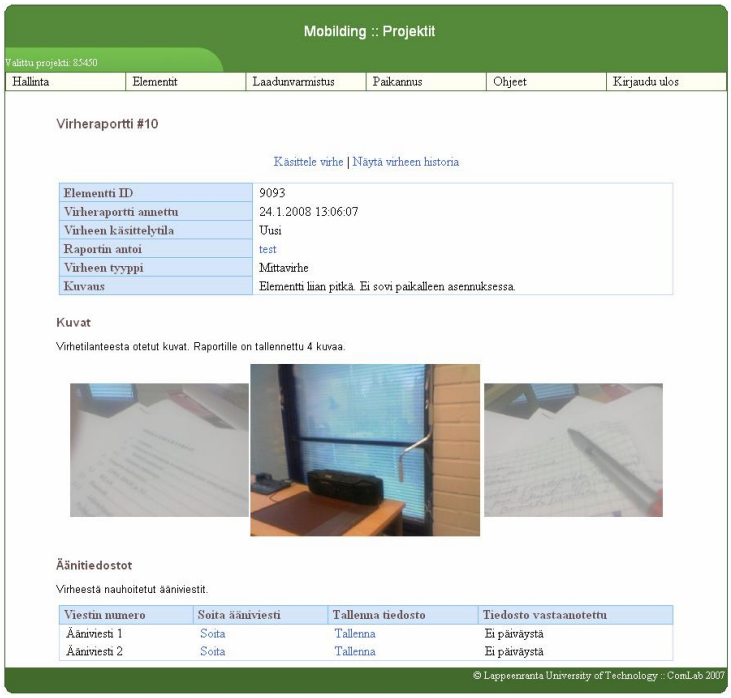
Käyttäjä voi aiheuttaa virheen kirjaamisen järjestelmään kahdella tavalla: joko syöttää

virheet suoraan sille tarkoitetulla Web Service liittymällä tai aiheuttaa virhe reaktiona mittatietoihin tarkistamisessa. Molemmissa tapauksissa virheet kirjataan tietokantaan. Tietoja ei koskaan kirjoiteta yli, vaan kantaan jää historia kaikista muutoksista. Virheraportti sisältää virheen kuvauksen tekstinä ja virheenkäsittelytilan. Virheen tyypit on määritelty kiinteästi järjestelmään haastatteluissa saatujen tietojen perusteella (luku 4). Nämä virhetyypit ovat nähtävillä taulukosta 6.1.

Virheen tunnus	Virhe	Kuvaus
1	Pintavaurio	Elementin pinnassa on vaurio tai värivirhe.
2	Mittavaurio	Elementin mitoissa on virhe
3	Vaurio	Osittainen hajoaminen; Elementti on fyysisesti vaurioitunut
4	Varausvirhe	Virhe elementin varausten paikoissa tai koossa
5	Hajonnut	Elementti on käyttökelvoton

Taulukko 6.1. Tietojärjestelmän virhetyypit

Virheraporttiin on myös mahdollista sitoa tiedostoja. Tämä on lähinnä tarkoitettu Web Service-rajapinnan kautta lähetettäville virheraporteille. Tätä kautta on mahdollista liittää virhekuvaukseen esimerkiksi kuva vioittuneesta elementistä. Järjestelmä itsessään ei ota kantaa, millaisia tiedostotyyppejä sinne lähetetään. Näin virheraporttiin voidaan liittää ongelmitta esimerkiksi äänitiedostoja tai mitä vain tulevaisuudessa tullaan katsomaan tarpeelliseksi. Ainoa ero on käyttöliittymän puolella, jossa liitetyt kuvatiedostot erotellaan omaan galleriaansa. Tämä on nähtävillä kuvasta 6.2, jossa on esimerkki järjestelmän graafisesta virheraportista.



Kuva 6.2. Graafinen virheraportti

Jokaiselle virheraportille tallennetaan tämän lisäksi käsittelyhistoria ja raportin tila. Tänne kirjataan kaikki muutokset mitä virheraportille on tehty ja onko elementti esimerkiksi korjattu. Käsittelyhistorian avulla voidaan jälkeenpäin kartoittaa, minkälaisia toimenpiteitä virheen korjaamiseksi tehtiin ja kuka toimenpiteet suoritti.

6.3.2 Mittatietojen tarkistaminen

Mittatiedot tarkastetaan niiden tietokantaan kirjaamisen yhteydessä. Elementeistä vastaanotetaan neljän tyyppistä mittatietoa: pituus, korkeus, leveys ja ristimitta. Tarkistaminen suoritetaan vertaamalla näitä toleranssirajoihin. Rajoina käytetään rakennusteollisuuden määrittämiä toleransseja, jotka ovat saatavilla kirjasta Betonielementtien toleranssit /BET03/. Näitä käytetään ainakin Joutsenon Elementti Oy:ssä toleranssirajoina. Valmistajalla on myös mahdollisuus määrittää omat toleranssirajat järjestelmään.

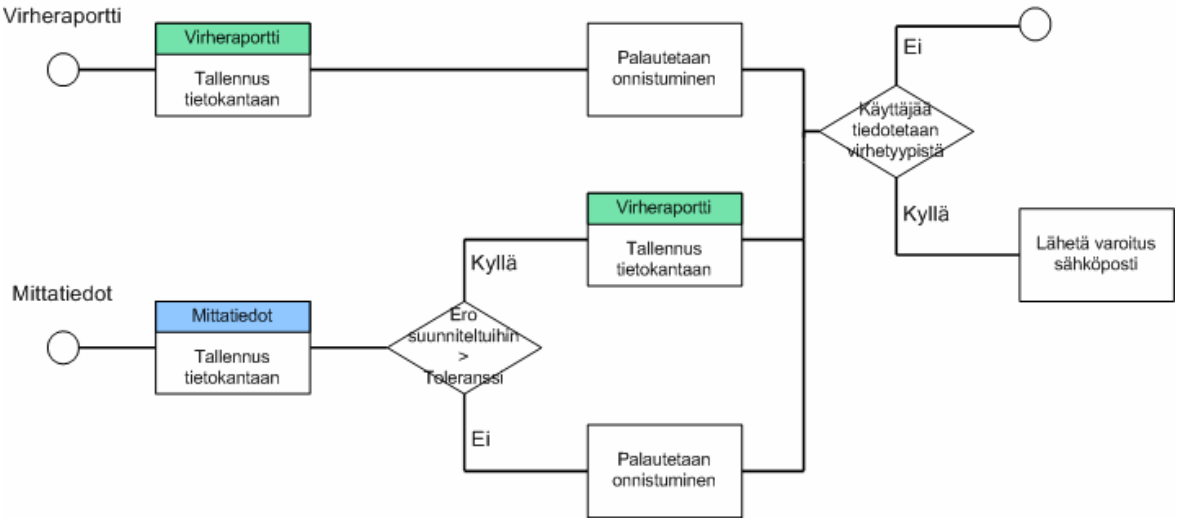
Kirjan perusteella elementeille määriteltiin kaksi toleranssirajaa: kiinteä määritys millimetreissä ja suhteellinen määritys prosenteissa. Näistä lukuarvoista käytetään aina

suurempaa vaihtoehtoa Betonielementtien toleranssit-kirjan ohjeistuksen mukaisesti. Vertaus suoritetaan elementin tyypin perusteella. Jokaisella elementtityypillä on omat toleranssirajansa ja tietojärjestelmään on määritelty oletusarvot jokaiselle pilottihankkeissa käytetylle tyypille.

Arvoja verratessa tarkastetaan ensin, onko tietokantaan tallennettu yrityksen omia toleranssirajoja. Niiden käyttö on aina etusijalla; jos yritys on määritellyt omat arvonsa, niitä käytetään tarkastuksissa. Jos yrityskohtaisia arvoja käytetään, toleranssirajat kaikille tarkistettaville dimensioille on syötettävä järjestelmään: oletusarvoja ja yrityksen omien arvoja ei yhdistellä automaattisesti. Jos tietyn dimension toleranssirajaa ei syötetä, järjestelmä ei tarkista kyseistä dimensiota saadessaan mittaustulokset, mutta tarkistaa edelleen ne, joille arvot on syötetty.

6.3.3 Reagointi

Tapahtumaketju virheraportin lähettämisessä ja mittatietojen tarkastamisessa on hyvin samankaltainen. Kuten kuvasta 6.3 nähdään, virheeseen reagoiminen voi alkaa joko virheraportin lähettämisestä tai mittatietojen lähettämisestä. Kummassakin tapauksessa matkapuhelimella otetaan yhteys Mobilding-palvelimen Web Serviceen ja lähetetään tiedot sinne. Ensimmäinen askel on aina tietojen tallentaminen tietokantaan. Arvoja ei kirjoiteta yli, vaan kaikista syötetyistä mittatiedoista jää merkintä tietokantaan. Järjestelmä olettaa, että uusin mittausarvo kertoo elementin voimassaolevat mitatut dimensiot.



Kuva 6.3. Virheiden käsittely järjestelmässä

Virheraportin tapauksessa virheraportti tallennetaan suoraan, mutta mittatietojen kohdalla aluksi tietokantaan tallennetaan vain mittatiedot. Tämän jälkeen mittatietoja verrataan suunniteltuihin dimensioihin ja tietokannan toleranssiarvoihin. Mikäli poikkeama ei ole sallituissa rajoissa, tallennetaan myös siitä virheraportti. Tämän jälkeen kummassakin tapahtumaketjussa palautetaan matkapuhelimelle viesti kirjauksen onnistumisesta. Jos mittatietietojen tarkistamisessa ilmenee virhe, lähetetään matkapuhelimeen siitä paluuviestissä tieto. Näin mittaaja saa puhelimeensa heti mittauksen yhteydessä hälytyksen, jos tulokset ylittivät toleranssirajat.

Matkapuhelimen kannalta tapahtumaketju on tässä vaiheessa ohi. Seuraavaksi palvelin tarkastaa kenelle tässä projektissa lähetetään varoituksia ilmenneistä virhetyypeistä. Jos yhteystietoja löytyy, lähettää järjestelmä sähköpostiosoitteisiin varoituksen tapahtuneesta.

6.3.4 Elementtien tilojen muutokset

Jokaiselle elementille on mahdollisuus määrittää tietojärjestelmässä tiloja. Tilojen käyttö on oleellinen asia laadunvarmistuksen käsittelyssä, sillä niiden avulla voidaan havaita poikkeamia aikataulutuksessa ja viestittää oleellista tietoa osapuolille. Tilat voivat olla kahdentyyppisiä: nykyisiä tiloja (joihin viitataan tästä lähtien tilana) tai suunniteltuja tiloja. Tilat kertovat mikä elementin tila on kysytyllä hetkellä ja suunnitellut tilat kertovat mikä

elementin tilan tulisi olla tietyllä ajankohdalla, esimerkiksi milloin elementin tulisi olla asennettu. Diplomityön toteutuksessa käytettiin elementin tiloina Tekla Structures-ohjelmiston tukemia elementin tiloja, sillä alun perin elementtien tiedot tuotiin järjestelmään tästä ohjelmistosta. Käytetyt tilat on listattu diplomityön liitteessä 4.

Suunniteltujen tilojen käyttäminen vaatii, että ne on syötetty järjestelmään elementtikohtaisesti. Käytännössä kyse on siis aikataulutuksen syöttämisestä järjestelmään. Tämä tulisi tehdä, kun elementit ensimmäistä kertaa lisätään Mobilding-järjestelmään. Kyseisten tietojen tuominen helpottuu, jos suunnitelmat on tallennettu jossain helposti käsiteltävässä muodossa: esimerkiksi tuominen Tekla Structures-ohjelmistosta onnistuu, mutta myös tietojen tuominen esimerkiksi Excel-tiedostosta on täysin mahdollista. Suunniteltuja tiloja voidaan tämän jälkeen automaattisesti vertailla elementtien tiloihin ja saavuttaa automaattiset aikataulutarkistukset ja varmistukset. Näin voidaan varmistaa, että valmistus ja rakentaminen etenevät suunnitelman mukaan.

Vaatimuksena kaikelle tilatietojen hyödyntämiselle on, että niitä päivitetään jatkuvasti ja aktiivisesti tapahtumaketjun eri osissa; elementin valmistuksesta pystytykseen. Kun elementit lisätään tietojärjestelmään, niille on jos siinä vaiheessa liitettävä tilatieto. Esimerkiksi yhdistäessä yrityksen elementti Mobilding järjestelmään, annetaan sille tila ”scheduled”, eli elementti odottaa valmistusta. Tämän jälkeen elementin tilaa päivitetään lähinnä matkapuhelimen ohjelmistolla. Elementin valmistumisen jälkeen siihen upotettu RFID-tunniste luetaan matkapuhelimella ja elementti kuitataan valmistuneeksi. Samalla periaatteella tilaa päivitetään myöhemmässä vaiheessa tapahtumaketjua; elementin tunniste luetaan ja elementille annetaan uusi tila. Operaatio on nopea ja on osapuolten tarpeesta ja harkinnasta kiinni, kuinka usein tilaa päivitetään. Jo muutaman tilan seuraamisella voidaan saavuttaa hyötyjä. Esimerkiksi kuittaamalla elementti valmistuneeksi tehtaalla, voidaan välittää tietoa rakennustyömaalle elementtien valmistumisesta ja kuittaamalla elementti asennetuksi voidaan seurata rakennuksen edistymistä ja välittää tästä tietoa osapuolille.

On huomattava, että tilojen muutoksia voidaan sisällyttää näkymättömästi toimenpiteisiin, jotka tehtäisiin muutenkin. Näin itse tilojen syöttämisestä ei välttämättä synny mitään

ylimääräistä työtä. Esimerkiksi tehtaalla suoritettavien laadunvarmistusmittausten yhteydessä voidaan elementin tila samalla muuttaa valmistuneeksi. Jos kaikkien elementtien mitat tarkastetaan joka tapauksessa, saadaan tilojen päivitys sisällytettyä samaan toimenpiteeseen. Jos mittojen tarkastuksessa ilmenee poikkeama, ei elementin tilaa tarvitse muuttaa. Näin ainoastaan tarkistetut, laatukriteerit täyttävät elementit päivittyvät tietojärjestelmään, josta eri osapuolet voivat nähdä ne. Tulevaisuudessa on myös mahdollista kehittää järjestelmää myös niin, että kuitataan esimerkiksi elementtilava tai yksittäinen kuljetus rakennustyömaalle saapuneeksi. Näin kaikki lavaan tai kuljetukseen yhdistetyt elementit päivittyisivät yhdellä operaatiolla, eikä siitä jälleen syntyisi ylimääräistä työtä.

Tilojen muutokset on parempi tehdä päätelaitteen kautta, eikä tehdä niistä automaattisia toimenpiteitä tietojärjestelmässä. Näin tehdyt muutokset pysyvät helpommin hallittavina, eikä järjestelmän logiikka tehdä tarpeettoman monimutkaiseksi. Esimerkiksi elementin mitat voidaan tarkistaa sekä tehtaalla sisäisessä laadunvarmistuksessa, että rakennustyömaalla. Tästä syystä emme voi määrittää järjestelmään yksinkertaisesti tilaa muuttumaan automaattisesti: elementin tila ei saa enää rakennustyömaalla muuttua takaisin valmistuneeksi, mikä ilmaisisi että se on yhä tehtaalla. Päivityksen yhteydessä voidaan käyttäjätunnusten oikeuksien perusteella tunnistaa käyttäjä ja päätellä tietojärjestelmään tehtävä muutos. Tällä hetkellä on kuitenkin yksinkertaisempaa määrittää tehtävät muutokset käytettävään matkapuhelimen ohjelmistoon.

Tilojen käyttämisellä saavutetaan elementtien tilan seuraaminen reaaliajassa. Esimerkiksi elementtien valmistaja näkee elementtien tilan jatkuvasti WWW-käyttöliittymän kautta ja tämä tieto voidaan jakaa myös muille osapuolille. Esimerkiksi rakennustyömaa tai kuljetusyritys näkee oman käyttöliittymänsä kautta tarkalleen, mitä elementtejä tehtaalla on valmiina kyseisellä hetkellä. Rakennustyömaat näkevät heti, milloin elementti on kuitattu valmistuneeksi ja milloin he voivat odottaa tarvitsemaansa elementtiä. Vastaavasti myös kuljetusyritykset näkevät mikä tilanne on tehtaalla ja milloin kuljetus on mahdollista hakea. Tapauksesta voi luonnollisesti myös informoida sähköpostiviestillä, mutta tämä ei vaikuta järkevältä ratkaisulta pelkästään jo syntyvän viestiliikenteen määrän vuoksi.

Tilatiedon jakaminen tuo kuitenkin läpinäkyvyyttä prosessiin: osapuolten on helpompi nähdä, mikä tilanne on esimerkiksi elementtitehtailla ja mitä on tarkoitus tapahtua tuotannossa lähitulevaisuudessa, jolloin tapahtumien ennakointi helpottuu.

Alun perin kaikki elementtitieto oli tarkoitus tuoda järjestelmään Tekla Structures virtuaalimallista ja tätä kokeiltiin käytännössä maaliskuussa 2007 Parma Oy:n kanssa toteutetussa pilottihankkeessa. Tässä operaatioissa elementeistä tuotiin järjestelmään niiden suunnitellut fyysiset ominaisuudet, sekä aikataulutus, eli elementin tulevat tilat. Pilotissa havaittiin kuitenkin kaksi ongelmaa, jotka vaikeuttivat testaamissa käytännössä: aikataulutusta ei käytetty pilottihankkeen mallissa ja ilmeni, ettei pienempien rakennusprojektien suunnittelussa välttämättä käytetä Teklan malliohjelmistoa. Tilatiedon liittäminen on suhteellisen tuore ominaisuus Tekla Structuresissa ja on epäselvää, miten paljon sitä rakennusteollisuudessa hyödynnetään. Toisekseen, käsiteltäessä tilannetta, jossa rakennussuunnitelmaa ei ole tehty mallinnusohjelmistolla, hankaloituu tilatietojen tuominen. Tällöin tiedot on joko syötettävä käsin tai on luotava yrityskohtainen ratkaisu, miten tiedot saadaan tuotua järjestelmään automaattisesti yrityksen suunnitelmista tai tuotannonohjausjärjestelmästä. Esimerkiksi Joutsenon Elementin tapauksessa elementit tuotiin järjestelmään parsimalla ne automaattisesti Excel-tiedostosta. Samaan dokumenttiin olisi mahdollista liittää aikataulutustietoa, jolloin ne voitaisiin automaattisesti lisätä järjestelmään.

6.3.5 Tilojen jatkuvat tarkistukset

Tietojärjestelmässä voidaan yhdistää sekä elementin tilat, että suunnitellut tilat ja vertailla näitä keskenään. Näin voidaan automaattisesti valvoa, että esimerkiksi elementtien asennus etenee aikataulun mukaisesti. Jos elementti on suunniteltu pystytettäväksi tiettynä päivänä, eikä sen tila muutu asennetuksi, voidaan tästä lähettää varoitus osapuolille (sekä valmistajalle, että rakennustyömaalle). Myös rakennustyömaan ja kuljetusliikkeiden näkökulmasta tämä olisi hyödyllistä. Rakennustyömaat tietävät aikataulun, millä he tarvitsevat elementit toimitetuksi. Automaattisilla tarkistuksilla voidaan ilmoittaa milloin elementit ovat valmistuneet tai varoittaa, jos elementtiä ei ole kuitattu valmistuneeksi

määräaikaan mennessä. Näin voidaan varmistaa, että kaikki elementit toimitetaan ajoissa ja varautua nopeasti tilanteisiin, joissa jotain poikkeavaa on tapahtunut.

Tekniseltä toteutukselta tarkistukset suoritetaan säännöllisesti ajettavalla cron-skriptillä. Ylläpidon kannalta ratkaisu monimutkaistaa tilannetta, sillä ohjelman ajaminen joudutaan määrittelemään erikseen itse tietojärjestelmän ulkopuolella. Tätä ei voi kuitenkaan välttää, sillä PHP-funktiot ajetaan vain niitä kutsuttaessa. MySQL-palvelimella on versiosta 5.1.6 alkaen ollut tuki New Event-toiminnalle /NEFM/. Tämä vastaa Linuxin cron-toiminnallisuutta, eli tietokantaan voidaan asettaa funktioita ajettavaksi itse määritettävin määräajoin. Toiminnallisuus ei valitettavasti auta tässä tilanteessa, sillä virheistä ilmoittaakseen järjestelmän tulee kyetä lähettämään siitä viestejä (sähköposti) ja tämä käytännössä vaatii PHP-funktion ajamista. Funktio tarkistaa elementtien tilan, vertaa niitä suunniteltuihin aikatauluihin ja reagoi määritetyllä tavalla. Tämä tarkoittaa, että käyttäjille lähetetään sähköposti-ilmoitus havaitusta ongelmasta elementtien tiloissa.

Riippuen elementtien määrästä, tämä toimenpide saattaa viedä hyvinkin paljon järjestelmän resursseja. Elementtien tiedot on käytävä läpi useampaan kertaan ja jokaiselle on suoritettava useita vertauksia. Jos elementtien määrä järjestelmässä on suuri, tämä saattaa viedä aikaa. Suoritus saattaa silti olla siedettävissä rajoissa, sillä tilat on merkitty tietorakenteeseen yhden päivän tarkkuudella, jolloin tarkistus täytyy suorittaa vain kerran vuorokaudessa. Se voidaan sijoittaa ajanjaksolle, jolloin järjestelmän kuormittaminen ei aiheuta ongelmia kenellekään käyttäjälle. Hakua voi myös algoritmisesti tehostaa, esimerkiksi karsimalla aluksi vain ne elementit, joille muutoksia on tulossa ja vertailemalla vain näiden elementtien tietoja.

6.4 Havaitut haasteet

Seuraavissa kappaleissa keskustellaan laadunvarmistuksen toteutuksessa ilmenneistä haasteista. Suuren haasteen muodostaa järjestelmän soveltaminen reaalimaailman tilanteisiin ja tästä aiheutuvat ongelmatilanteet. Koko järjestelmän tarkoitus on tehostaa työntekoa, säästää aikaa ja pienentää kustannuksia. Täten sen käyttö ei saisi aiheuttaa

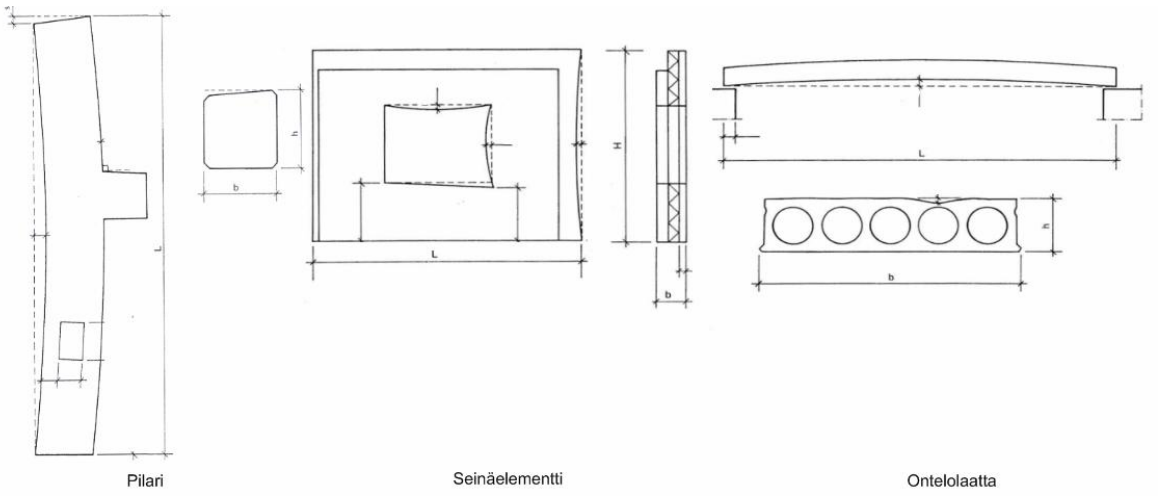
ylimääristä työtä ja viestiminen tulisi olla hyödyllistä ja nopeaa.

Suurimpina haasteina kappaleissa nostetaan esille elementtien mittauksien ongelmallisuus ja toipuminen tilanteista, jossa tietojärjestelmän tilanne ei vastaa enää reaalimaailman tilannetta. Myös poikkeuksista viestimiseen on kiinnitettävä huomiota. Tällä hetkellä kommunikaatio tapahtuu sähköpostin välityksellä ja kriittisen viestintään se ei välttämättä ole sopivin vaihtoehto.

6.4.1 Haasteet mittauksen ja tietojärjestelmän kannalta

Elementtien mittaaminen käytännön olosuhteissa aiheuttaa haasteita, kun tilannetta ajattelee tietojärjestelmän kannalta. Tietojärjestelmässä elementtien suunnitellut mitat tuodaan järjestelmään oletustilanteessa virtuaalimallista (Tekla Structures) ja järjestelmän muuttujat on suunniteltu tämän pohjalta. Näin jokaisella elementillä on selkeästi määritellyt dimensiot korkeus, leveys ja paksuus.

Reaalimaailman mittauksissa ilmenee kuitenkin haasteita: elementit eivät ole välttämättä symmetrisiä tai identtisiä. Mitatessa esimerkiksi pilaria, dimensiot on helppo mitata käytännön olosuhteissa. Sen dimensiot ovat selkeät ja pilari on muodoltaan symmetrinen, joten on helppo saada mitat sijainneista, jotka vastaavat teoreettisia suunniteltuja mittoja. Mitatessa kuitenkin esimerkiksi seinäelementtejä, tilanne voi olla toinen. Tämän tyyppisessä elementissä osa sivustoista voi olla kaltevia ja esimerkiksi elementin leveys saattaa vaihdella riippuen mistä kohtaa elementin sivustaa mittauksen suorittaa. Teoreettisissa mitoissa leveys tai korkeus on kyseisen dimension maksimiarvo elementille, mutta tämän mittaaminen käytännön olosuhteissa saattaa olla hyvin ongelmallista. Mittaukset suoritetaan käsikäyttöisellä laser-mittarilla, joka on yhdistetty matkapuhelimeen Bluetooth-yhteyden välityksellä. Jos mittaajan on huolehdittava hankalista mittausolosuhteista (suuret korkeudet tai etäisyydet) tai esimerkiksi juuri kaltevista pinnoista, jolloin on hankala tarkasti asettaa mittaria elementtiin, mittaustulosten laatu kärsii varmasti. Tulosten lisäksi mittauksesta tulee myös vaivalloisempaa ja juuri tätä yritetään välttää mittauksen automatisoimisella.



Kuva 6.4. Elementtityypit ja eroavat dimensiot /BET03/

Ristiriidat elementtien dimensioissa muodostavat myös ongelman. Käsiteltävät elementit eivät ole samanlaisia ja osalla on erilaisia dimensioita. Esimerkkinä voidaan verrata vaikka kolmea elementtityyppiä, jotka ovat nähtävillä myös kuvassa 6.4: seinäelementit, pilarit ja ontelolaatat. Näillä kolmella tyypillä on toisistaan eroavia mitattavia dimensioita. Osittain erot ovat semanttisia, esimerkiksi pilareilla suurin dimensio on pituus, mutta seinäelementeillä tätä dimensiota ei ole lainkaan, vaan puhutaan paksuudesta. Tietorakenteen ongelmat ovat lähtöisin alkulähtökohdasta, jolloin elementtien mittatiedot tuotiin järjestelmään Teklan mallista. Täällä kaikille elementeille oli määritelty vain pituus, korkeus ja leveys. Elementtien ominaisuudet on silti pystyttävä kartoittamaan tietokantaan yksikäsitteisesti. Jos tulevaisuudessa järjestelmässä käsitellään vielä teräselementtejä, ongelma korostuu, sillä teräselementtien mitatut ominaisuudet eroavat vielä enemmän betonielementeistä. On haasteellista esittää eriävien elementtien rakennetta tietojärjestelmässä yhtenevällä tavalla, säilyttäen yhä tietojen helppo analysointi.

Kiinnikkeiden paikat ja reiät elementeissä ovat myös ongelman lähde. Projektin sidosryhmät ilmaisivat kiinnostuksen myös varmistaa elementeissä olevien kiinnikkeiden sijainteja mittauksella. Ongelma näissä on, että ne ovat täysin rakennussuunnitelmakohtaisia. Toleranssien määrittäminen päädimensioille (kuten korkeus ja leveys) voidaan asettaa helposti, koska ulkoisesti elementit ovat samankaltaisia.

Elementteihin lisättävät reiät, ripustukset ja kiinnikkeet sen sijaan ovat täysin tapauskohtaisia; ne voivat sijaita eri paikoissa elementistä elementtiin, vaikka päädimensiot olisivatkin samat. Betonielementtien toleranssit-kirjassa näille myös määritellään toleranssirajat, mutta itse mittaaminen käytännön tilanteessa on hyvin haastavaa. Tällöin tulisi tarkasti määrittää tyyppikohtaisesti, mistä mittaukset tehdään ja tämä on enemmänkin haaste puhelimen käyttöliittymälle, joka on apuna itse mittauksessa. Tietojärjestelmän kannalta elementeille voidaan määrittää n-kappaletta ylimääräisiä mitattavia objekteja omilla toleranssirajoillaan, mutta niiden fyysistä sijaintia elementeissä on hyvin vaikea määrittää. Käytännön mittauksissa täytyy olla käytössä sovittu protokolla, jolloin kaikki elementit mitataan samalla tavalla.

6.4.2 Reaalimaailman ongelmista toipuminen

Jotta järjestelmä olisi hyödyllinen, sen tulee sopeutua tilanteisiin, jossa järjestelmän tilanne ei vastaa enää reaalimaailman tilannetta. Tällainen tilanne tulee väistämättä jossain vaiheessa eteen: elementti voidaan esimerkiksi unohtaa kuitata valmistuneeksi tai tietoliikenneyhteydet eivät toimi kuittaushetkellä. Tällöin elementin tilat eivät vastaa järjestelmän näkemää tilannetta: elementti voi olla asennettu, mutta järjestelmä lähettää silti varoituksia luullessaan, että elementti on yhä tehtaalla.

Tällaisessa tilanteessa yhtenä vaarana on turhien varoitusten jatkuva lähettäminen. Jos varoitukset eivät ole tarkkoja, käyttäjät lopettavat niiden seuraamisen ja niistä tulee hyödyttömiä. Järjestelmään tulee tarjota mahdollisuus muuttaa elementtien tiloja ja tietoja WWW-käyttöliittymän kautta ja tarjota mahdollisuus lopettaa virheilmoitusten lähettäminen tietyksi määräajaksi. Näin olematon virhe ei häiritsisi käyttäjiä turhaan.

Järjestelmän kannalta tilojen vaihdolle voitaisiin määrittää myös tietty aikaikkuna. Tällä tarkoitetaan, että tietyn tilan jälkeen odotetaan, että määritetyn ajan jälkeen tapahtuu tiettyjä tapahtumia tai tiloja. Esimerkiksi jos elementin tilaksi on määritetty, että sitä kuljetetaan rakennustyömaalle, tiedetään kohtuullisella varmuudella sen saapuvan rakennustyömaalle saman päivän aikana. Järjestelmä voidaan määrittää muuttamaan tila

automaattisesti, vaikka kuittausta ei olisi tapahtunut tai huomauttamaan asiasta vastuussa olevaa käyttäjää.

6.4.3 Virheiden priorisointi ja viestintä

Tällä hetkellä järjestelmä kohtelee kaikkia virheitä samanarvoisina. Ainoana erona on, että käyttäjä voi valita minkä tyyppisistä virheistä häntä tiedotetaan. Virheet tulisi kuitenkin priorisoida ja selvittää, mitkä niistä ovat kriittisimmät. Kriittiset virheet ovat virheitä, jotka aiheuttavat eniten kustannuksia ja jotka täytyy korjata heti. Luvussa 2 nykytilan selvityksessä kävi ilmi, että tällä hetkellä pienimmät virheet käydään kootusti läpi vasta projektin lopussa. Vaikka näistä on hyvä tiedottaa etukäteen elementtitehdasta, ei niitä silti pitäisi sotkea vakavampien virheiden sekaan.

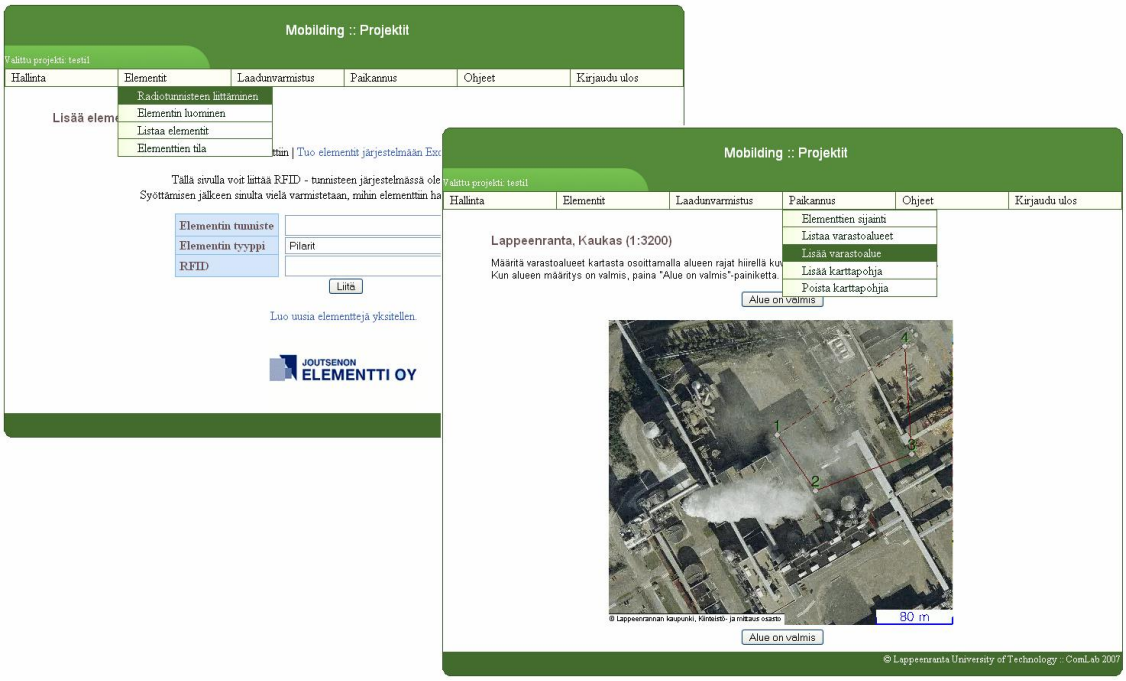
Tämä nostaa esiin kysymyksen, kuinka virheistä tulisi viestittää käyttäjiä. Sähköposti ei sovellu kriittiseen viestintään, jolloin tieto täytyisi saada vastuuhenkilöille mahdollisimman reaaliajassa. Jos kaikista virheistä viestitään sähköpostilla, se aiheuttaa helposti liikaa viestimistä. Tärkeät viestit hukkuvat helposti ja viestinnän hyödyllisyys katoaa. Virheistä tulisi viestiä niiden tärkeyden mukaan ja käyttäen siihen soveltuvaa viestintäkeinoa.

Kriittisiin viesteihin yksi vaihtoehto voisi olla SMS-viestit (Short Message Service). Useimmat ihmiset kantavat matkapuhelinta mukanaan ja lähettämällä siihen tekstiviesti, saavutetaan huomattavasti nopeampi tapa kommunikoida poikkeamasta. SMS-järjestelmän liittäminen Mobilding-järjestelmään olisi myös suhteellisen vaivatonta.

Pienistä virheistä voisi viestiä kokoamalla useita ilmoituksia säännöllisesti lähetettävään sähköpostiin tai esimerkiksi ilmoittamalla niistä vain WWW-käyttöliittymän kautta. Järjestelmä voisi sisäänkirjautumisen jälkeen ilmoittaa kuinka monta uutta raporttia on kirjattu edellisen kirjautumisen jälkeen ja luokitella ne valmiiksi prioriteettien mukaan.

6.5 Graafinen käyttöliittymä

Järjestelmän hallinta tapahtuu graafisen käyttöliittymän kautta. Käyttöliittymä on toteutettu WWW-sivuina, jolloin käyttö ei vaadi ylimääräisten ohjelmistojen asentamista käyttäjille, vaan käyttö onnistuu WWW-selaimella. Käyttöliittymä sisältää useita näkymiä eri käyttäjäryhmille ja tarjoaa jokaiselle käyttäjäryhmälle tarpeelliseksi katsotun toiminnallisuuden projektin tietojen tarkasteluun ja hallintaan. Käyttöliittymä pyrittiin pitämään selkeänä ja yksinkertaisena käyttää. Esimerkki kahdesta käyttöliittymän sivusta on nähtävillä kuvasta 6.5.



Kuva 6.5. Esimerkki Mobilding-järjestelmän käyttöliittymästä

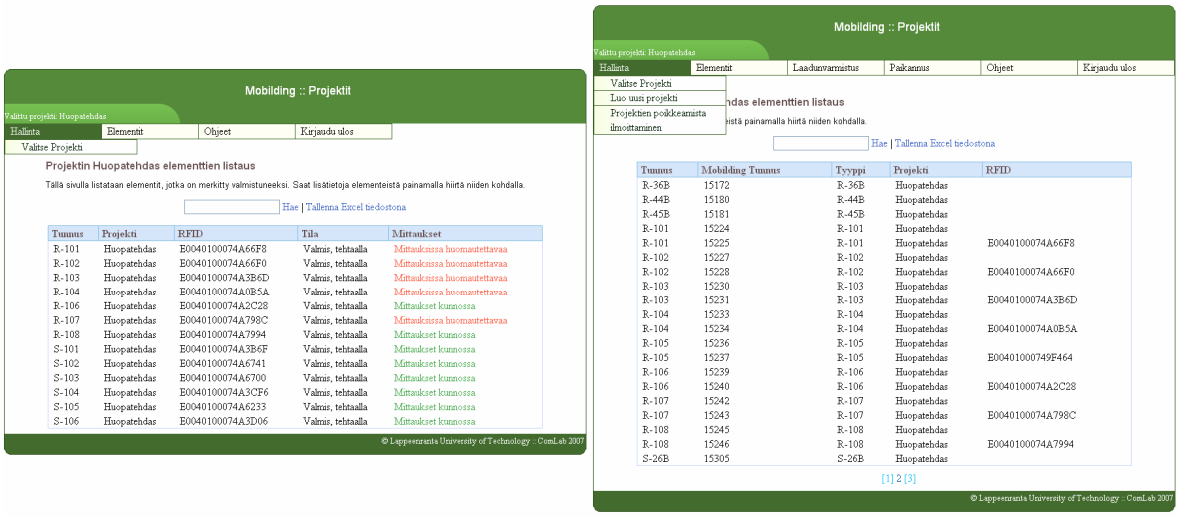
Graafinen ilme on luotu pääasiassa CSS-määritysten (Cascading Style Sheets) avulla. Sivuille liitettävän grafiikan (kuvat) avulla on pääasiassa pyritty vain selkeyttämään valintoja. Näin on tehty esimerkiksi määriteltäessä kartta-alueita tai toleranssiparametrejä, jolloin näytetään kuva rakennuselementin dimensioista. Mikäli ulkoasua halutaan kehittää eteenpäin, muokkaaminen on helppoa, sillä ohjelmakoodi on eroteltu ulkoasusta. Vaikka sivut ovat käytännössä HTML-kieltä, mukaan on liitetty niin paljon JavaScript-kieltä, että

tätä tukevien selainten käyttäminen on välttämätöntä. JavaScript-kielen käytöllä pyrittiin parantamaan sivujen käytettävyyttä ja dynaamisuutta, sekä ylipääsemään CSS-toteutuksen eroja eri selainten välillä, erityisesti Internet Explorer-selaimen tapauksessa.

6.5.1 Näkymät ja rakenne

Käyttöliittymä jakaantuu näkymiin, jotka eroavat käyttäjäryhmien välillä. Eri käyttäjien tarkastellessa samaa sivua, heille esitetty näkymä riippuu tunnuksista, joilla he ovat kirjautuneet sisään. Määrittäviä tekijöitä näkymässä ovat yritys, johon käyttäjä on yhdistetty ja yrityksen rooli Mobilding-järjestelmässä. Mahdollisia yrityksen rooleja järjestelmässä ovat valmistaja, kuljetus ja rakentaja.

Alun perin diplomityössä oli tarkoitus luoda käyttöliittymä kaikille kolmelle osapuolelle, mutta diplomityön aikana toteutettujen pilottihankkeiden ja yhteistyökumppaneiden myötä katsottiin kuitenkin tarpeelliseksi luoda käyttöliittymät vain rakennustyömaalle ja valmistajalle. Perusnäkymä, joka käyttäjälle aukeaa, riippuu tästä yrityksen roolista, jonka alle käyttäjä on sidottu. Kuvassa 6.6 verrataan rakennustyömaan ja valmistajan näkymiä kun molemmilla on avattuna sama sivu, jossa listataan valitun projektin elementit. Kuten kuvasta on nähtävillä, rakennustyömaan navigointivalinnat ovat huomattavasti suppeammat ja tarjolla on rajoitettu määrä funktioita. Myös elementtien listauksessa on eroja. Esimerkkisivulla rakennustyömaan näkymässä esitetään vain valmistuneet elementit, kun valmistaja puolestaan näkee kaikki heiltä tilatut elementit. Toteutuksessa rakennustyömaalle katsottiin oleellisimmaksi tiedoksi elementtien valmistumisen ajankohta ja laadunvarmistusmittausten onnistuminen. Näkymän kautta pystytään myös tarkastelemaan elementtien yksityiskohtaisia tietoja, mutta muu toiminnallisuus on piilotettu.



Rakennustyömaa Valmistaja

Kuva 6.6. Kahden käyttäjäryhmän näkymä samasta sivusta

Perusnäkymissä voi myös olla yrityskohtaisia eroja. Pilottihankkeissa oli mukana kaksi eri valmistajaa: Parma Oy ja Joutsenon Elementti Oy. Yritysten kirjanpidosta ja rakennussuunnitelmien eroista johtuen RFID-tunnisteiden syöttäminen kummankin yrityksen tapauksessa täytyi hoitaa eri tavalla. Nyt käyttäjän kirjautuessa Parma Oy:n tai Joutsenon Elementti Oy:n tunnuksilla järjestelmään, on näytettävä näkymä tunnisteiden syöttämisestä erilainen, vaikka navigointivalinnat ja muu sivujen rakenne pysyykin samanlaisena. Parma Oy:n tapauksessa näkymiin liitetään ohjaavia kuvia, joiden avulla työntekijä osaa syöttää oikeat arvot yrityksen kirjanpidosta järjestelmään. Joutsenon Elementti Oy:n tapauksessa elementtejä syötettäessä tarjotaan mahdollisuus tuoda tiedot järjestelmään suoraan Microsoft Excel-tiedostosta. Nämä erot ovat täysin yrityskohtaisia: niiden näyttämisestä toisen yrityksen käyttäjille ei ollut mitään lisäarvoa ja se saattaisi vain aiheuttaa sekaannusta. Tästä syystä toiminnallisuus piilotettiin näkymiin, jotka ovat vain niitä tarvitsevien käyttäjien saatavilla.

6.5.2 Käyttöliittymän toiminnallisuus

Käyttöliittymän toiminnallisuus vaihtelee kirjautuneen käyttäjän tunnuksista, kuten on käynyt ilmi aikaisemmista luvuista. Pääosa toiminnallisuudesta on toteutettu elementtivalmistajien näkymään käyttöliittymästä, sillä kuten luvun 2 taustatutkimuksessa

saatiin selville, he tekevät suurimman osan päätöksistä mitä vioittuneille elementeille tehdään. Elementtitehtaat myös toimivat aktiivisina yhteistyökumppaneina pilottihankkeissa, jolloin toiminnallisuutta pystyttiin testaamaan ja kehittämään. Rakennustyömaan näkymä käyttöliittymästä sisältää toiminnallisuuden elementtien tietojen tarkastamiseen, kattaen tärkeimpänä niiden tilatiedon ja laadunvarmistustiedon. Näkymästä ei kuitenkaan voi muokata järjestelmän tietoa.

Hallinta on toteutettu elementtivalmistajan käyttöliittymän puolelle. Näiden sivujen toiminnallisuus on esitelty taulukossa 6.3. Normaalissa käyttötapauksessa valmistaja määrittää aluksi pilottiprojektiin aliprojektin, joka käytännössä vastaa valmistajan kirjanpidossa olevaa tilausta kyseiseen pilottiprojektiin. Tämän jälkeen aliprojektin alle tuodaan rakennussuunnitelmasta elementit, jotka halutaan käsitellä. Ne voidaan syöttää käsin tai tuoda automaattisesti esimerkiksi Excel-tiedostosta tai Tekla Structures-rakennusmallista. Viimeinen vaihe on yhdistää elementteihin RFID-tunniste käyttöliittymän kautta.

Kun tiedot on syötetty järjestelmään, voidaan määrittää laadunvalvonta-asetukset. Käyttäjä voi määritellä minkä tyyppisistä virheistä järjestelmä tiedottaa ja kenelle virheistä ilmoitetaan. Elementtityypeille voidaan syöttää toleranssirajat, joita käytetään matkapuhelimella lähetettyjen mittausten tarkistamiseen. Tämän jälkeen järjestelmästä voidaan seurata sinne kirjattuja tuloksia.

Elementtien tiedot ja tilat muuttuvat havaittujen poikkeamien ja saatujen syötteiden mukaan. Käyttöliittymästä voidaan tarkastella listauksia projektien sisältämistä elementeistä, tarkastellen esimerkiksi tiloja tai mittaustuloksia. Saatuja virheraportteja voidaan hallita käyttöliittymän kautta. Ne tarjoavat graafiset raportit rekisteröidyistä virheistä ja mahdollistavat raporttien tilan muokkaamisen. Kaikki esitettävä tieto on haluttu tehdä mahdollisimman käytettäväksi ja tästä syystä listaukset on mahdollista tuoda ulos käyttöliittymän kautta Excel-yhteensopivina CSV-tiedostoina (Comma-separated values). Myös sivujen tulostukseen käytetään omia CSS-määrityksiä, jolloin tulosteista saadaan selkeitä.

Toiminto	Kuvaus	Muuta tietoa
Aliprojektien määrittely	Luo tilauksia tai aliprojekteja Mobilding-projektin alle	
Projektin valinta	Valitse käsiteltävä aliprojekti	
Virheistä tiedottaminen	Valitse yhteyshenkilöt ja virhetyypit joista lähetetään varoituksia	
Varastoalueiden määrittely	Määrittele varastoalueet graafisesti GPS-kartasta tietojärjestelmään	Valinta suoraan kartasta point&clik-periaatteella
Elementtien lisääminen ja yhdistäminen	Tuo elementtejä järjestelmään ja yhdistä ne RFID - tunnisteisiin	Mahdollisuus tuoda elementit järjestelmään Microsoft Excel-tiedostosta
Elementtien listaus	Listaa elementit eri parametrien mukaan	Mahdollisuus viedä elementit ulos järjestelmästä Excel-yhteensopivassa formaatissa
Virheiden hallinta	Tarkastele ja käsittele virheraportteja	Mahdollisuus viedä virheen tapahtumahistoria ulos järjestelmästä Excel-yhteensopivassa formaatissa
Laatuparametrien määritys	Määrittele laatuparametrit rakennuselementtityyppikohtaisesti	
Mittaustulosten listaus	Tarkastele elementeistä saatuja käytännön mittaustuloksia	

Taulukko 6.3. Käyttöliittymän toiminnallisuus

7. JOHTOPÄÄTÖKSET

Rakennusteollisuuden nykytilanteen kartoituksessa kävi ilmi, että diplomityössä toteutetulle laadunvarmennusjärjestelmälle on olemassa todellista tarvetta. Tuloksissa ilmeni, että varsinkin pienemmissä elementtitehtaissa laadunvarmistus ja tulosten kirjaaminen tehdään hyvin pitkälti käsityöllä. Käsityöllä tehtävän työn luonteesta johtuen valvonta aiheuttaa liikaa lisäkustannuksia, jotta sitä tehtäisiin tasaisesti jokaisessa tilauksessa ja suureen osaan toimitetuista elementeistä. Diplomityössä toteutettu järjestelmä automatisoi tätä prosessia ja tekee siitä kustannustehokasta, mahdollistaen tiedon helpon arkistoinnin ja vähentää valvontaan tarvittavia työtunteja.

Diplomityössä saatiin tuloksena toimiva laadunvarmistusjärjestelmä, joka toimii hyvänä perustana jatkokehitykselle. Järjestelmä mahdollistaa virheraportoinnin, rakennuselementtien tilatietojen seuraamisen ja poikkeustilanteista ilmoittamisen käyttäjille sähköpostin välityksellä. Järjestelmää hallitaan WWW-käyttöliittymän kautta, mutta siihen toteutettiin myös Web Service-rajapinta, joka mahdollistaa viestinnän mobiililaitteiden kanssa. Käytännön työtilanteessa työntekijät rakennustyömaalla tai elementtitehtaalla tunnistavatkin rakennuselementit matkapuhelimella ja käyttävät puhelinta kommunikoidakseen laadunvarmistusjärjestelmän kanssa.

Järjestelmää ehdittiin testata marraskuussa 2007 yhdessä pilottihankkeessa Parma Oy:n kanssa, jolloin testattiin RFID-tunnusten yhdistämistä tietojärjestelmän elementteihin. Hanke suoritettiin onnistuneesti: elementtitiedot tuotiin Mobilding-järjestelmään Tekla Structures-rakennusmallista ilman ongelmia ja RFID-tunnisteet yhdistettiin käyttöliittymän kautta oikeisiin rakennuselementteihin. Saatu palaute Parma Oy:ltä oli positiivista ja Mobilding-järjestelmän käyttöliittymä koettiin selkeäksi käyttää. Järjestelmälle on vielä tarkoitus toteuttaa kaksi pilottihanketta; ensimmäinen tammikuussa 2008 Joutsenon Elementti Oy:n kanssa testaamaan elementtien mittausta ja virheraportointia ja myöhemmin toinen Mikkelin Betoni Oy:n kanssa testaamaan järjestelmän karttapaikannusta GPS-koordinaattien perusteella. Kirjoitushetkellä näitä testihankkeita ei

vielä ollut ehditty toteuttaa, mutta järjestelmän teknisiä osa-alueita on testattu ja ne on havaittu toimiviksi.

Järjestelmän toteutuksella saavutettiin hyötyjä, joista erityisesti toimintojen ketjuttamisella voidaan tehdä prosessista tehokkaampaa ja nopeampaa. Tapahtumia voidaan näkymättömästi yhdistää toisiinsa ilman, että työntekijältä vaaditaan ylimääräistä työtä. Esimerkiksi mittatietojen tarkastuksen yhteydessä tulokset automaattisesti kirjautuvat tietojärjestelmään ja elementin tila järjestelmässä muutetaan valmistuneeksi. Näin rakennustyömaa näkee reaaliajassa käyttöliittymästään milloin elementit ovat valmiina tehtaalla ja laatutiedot ovat välittömästi valmistaja käytätettävissä helposti käsiteltävässä muodossa.

Mittaustulosten automaattisella kirjaamisella voidaan myös mahdollisesti saavuttaa kustannussäästöjä. Valmistajille tehdyistä haastatteluista kävi ilmi, että mittaustulosten säilyttäminen ja arkistointi on puutteellista, vaikka niitä suoritettaisiinkin. Tulokset usein säilötään mappeihin tai ne hylätään, kun tilaus on toimitettu. Tallentamalla tulokset automaattisesti tietojärjestelmään mahdollistuu myös niiden helppo analysointi. Tuloksista voidaan tarkastella, millaista hajontaa laadussa on esimerkiksi elementtityyppi-, tilaus- tai päivämääräkohtaisesti. Näin voidaan kartoittaa, esiintykö laadussa huononemista tiettyinä ajankohtina tai tietyissä elementtityypeissä. Tiedon avulla voidaan kartoittaa syy-seuraus suhteita ja saavuttaa konkreettisia parannuksia toimitusketjussa. Koska tietojen syöttämiseen käytetty matkapuhelinohjelmisto vaatii sisäänkirjautumisen, voidaan tietoja katsella myös käyttäjä/työntekijäkohtaisesti.

Jatkokehityksessä laadunvalvontajärjestelmän kannalta tulee kiinnittää huomiota virheiden priorisointiin ja viestintäkeinoihin. Tällä hetkellä kaikkia virheitä kohdellaan samanarvoisina ja niistä ilmoitetaan samalla tavalla. Todellisuudessa jotkin virhetyypit ovat huomattavasti vakavampia ja voivat jopa pysäyttää rakentamisen etenemisen. Näistä tulisi viestiä mahdollisimman nopeasti ja niin, että viesti myös huomataan ajoissa. Sähköposti ei tähän sovellu, mutta yksi vaihtoehto olisi SMS-järjestelmän käyttäminen. Tekstiviestillä ihmiset tavoitetaan huomattavasti nopeammin sijainnista riippumatta, sillä

matkapuhelin kulkee yleensä työntekijöiden mukana. Virhetyyppien analysointi ja vaihtoehtoisten viestintäteiden toteutus kuitenkin jäi tämän diplomityön ulkopuolelle.

Tekniseltä näkökulmalta rajapintojen toteutusta kannattaa tarkastella. Mikäli PHP:n sisäänrakennetussa SOAP-tuessa tapahtuu parannuksia, siihen siirtyminen saattaa olla kannattavaa yhteensopivuuden ja jatkokehityksen kannalta. Tässä diplomityössä valittu PEAR::SOAP-laajennus ei osoittautunut palvelinpään toteutuksen kannalta niin joustavaksi, kuin oli toivottu. Se on selkeästi vielä kehitystilassa ja kattavan dokumentaation puute tekee kehityksestä haastavaa. Koska laajennus huolehtii vain SOAP-viestinnästä, on toiminnallisuuden siirtämineen toiseen ratkaisuun helppoa, sillä toteutettuja funktioita ei tarvitse muuttaa.

LÄHTEET

[ALO04] Alonso, Gustavo. 2004. Web Services – Concepts, Architecture and Applications. ISBN 3-540-44008-9

[BET03] Betonielementtien toleranssit. 2003. Betonikeskus Ry. ISBN: 952-5075-53-2.

[BUI05] Buildercom. buildernews, 1/2005. [pdf-dokumentti] Viitattu: 3.12.2007 Saatavilla: http://www.buildercom.fi/data.php/200510/069710200510051209_buildernews_1_2005.pdf

[CHE06] Cheng, H.K. Tang, Q.C. Zhao, J.L. 2006. Web Services and Service-Oriented Application Provisioning: An Analytical Study of Application Service Strategies. Engineering Management, IEEE Transactions on.Volume: 53 , Issue: 4. Sivut: 520 - 533. ISSN: 0018-9391

[DAV02] Davis, D. Parashar, M.P. Latency Performance of SOAP Implementations. 2002. Cluster Computing and the Grid, Toukokuu / 2002. 2nd IEEE/ACM International Symposium on. Sivut: 407 – 407. Digital Object Identifier 10.1109/CCGRID.2002.1017169

[ERA06] Era Build. 2006. RFID in Construction, Final Report. Kesäkuu, 2006. [pdf-dokumentti] Viitattu: 28.11.2007. Saatavilla: http://www.formas.se/upload/RFID%20in%20Construction%20-%20Final%20report.pdf

[KAU07] Kauko, Juhani. 2007. Toimitusjohtaja, Joutsenon Elementti Oy. Puhelinhaastattelu 24.9.2007. Haastattelija: Reisbacka, Teemu. Haastattelumuistiinpanot haastattelijan hallussa.

[KA07B] Kauko, Juhani. 2007. Toimitusjohtaja, Joutsenon Elementti Oy. Puhelinhaastattelu 30.8.2007. Haastattelija: Reisbacka, Teemu. Haastattelumuistiinpanot haastattelijan hallussa.

[LGPL] GNU Lesser General Public License, Version 3. Päivitetty 29.6.2007. [www-dokumentti] Viitattu: 5.12.2007. Saatavilla: http://www.gnu.org/copyleft/lesser.html

[KOS07] Koskinen, Mikko. 2007. Yhteyshenkilö, Mobilding-projekti, Parma Oy. Sähköpostikirjeenvaihto, aiheena ”Mobilding hankkeesta kysymyksiä” 3.10.2007. Haastattelija: Reisbacka, Teemu. Sähköpostit haastattelijan hallussa.

[LES06] Leskinen, Sonja. Mobile Solutions and Construction Industry – Is it a working Combination? Gradutyö, VTT Publications. ISSN: 1235–0621

[NEFM] MySQL New Event Feature Manual. [www-dokumentti] Viitattu: 30.11.2007 Saatavilla: http://dev.mysql.com/tech-resources/articles/event-feature.html

[NUS07] NuSOAP-projektin sivut. 2007. Päivitetty 27.7.2005. [www-dokumentti] Viitattu: 29.11.2007. Saatavilla: http://sourceforge.net/forum/?group_id=57663

[OASIS] Oasis. UDDI Specification. [www-dokumentti] Viitattu: 18.6.2007. Saatavilla: http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm

[PEA07] PEAR Manual, päivitetty 25.11.2007. [www-dokumentti] Viitattu: 27.11.2007. Saatavilla: http://pear.php.net/manual/en/introduction.php

[PHA06] Phan, Khoi Anh. Tari, Zahir. Bertok, Peter. 2006. A benchmark on soap's transport protocols performance for mobile applications. Proceedings of the 2006 ACM symposium on Applied computing. Sivut: 1139 - 1144. ISBN:1-59593-108-2

[PHPL3] The PHP Group. The PHP License, versio 3.01. [www-dokumentti] Viitattu: 5.12.2007. Saatavilla: http://www.php.net/license/3_01.txt

[PHS07] PHP5 SOAP-sivusto. 2007. [www-dokumentti] Viitattu: 29.11.2007. Saatavailla: http://fi.php.net/soap

[REI07] Reisbacka, Teemu. 2007. Rajapinta Tekla Structures-ohjelmistoon. Kandidaatintyö, Lappeenrannan teknillinen yliopisto.

[RPC07] Cook, Charles. XML-RPC.NET. [www-dokumentti] Viitattu: 18.6.2007. Saatavilla: http://www.xml-rpc.net/

[SCH07] SOAP::PEAR projekti sivusto, päivitetty 29.6.2007. Viitattu: 27.11.2007. Saatavilla: http://pear.php.net/package/SOAP

[SEL06] Selfa, D.M. Carrillo, M. Del Rocio Boone, M. 2006. A Database and Web Application Based on MVC Architecture. Electronics, Communications and Computers, 2006. CONIELECOMP 2006. 16th International Conference. Digital Object Identifier 10.1109/CONIELECOMP.2006.6

[SER06] Sergey, Kim. 2006. Composing Web Services. Diplomityö, Lappeenrannan teknillinen yliopisto.

[SHA03] ShaikhAli, A. Rana, O.F. Al-Ali, R. Walker, D.W. 2003. UDDIe: an extended registry for Web services. Applications and the Internet Workshops, 2003. Proceedings. 2003 Symposium. Sivut 85 - 89. INSPEC Accession Number: 7685623.

[SMART] Smarty Template Engine Manual. 2007. Päivitetty: 17.9.2007. [www-dokumentti] Viitattu: 29.11.2007. Saatavilla: http://smarty.php.net/manual/en/

[SMAVA] Smarty Validate-plugin kotisivu. 2007. Päivitetty 23.4.2007 [www-dokumentti] Viitattu: 18.12.2007. Saatavilla: http://www.phpinsider.com/php/code/SmartyValidate/

[SUI07] Suikka, Arto. 2007. Tuoteryhmäpäällikkö (betonielementit), strategiset kehityshankkeet, Rakennustuoteteollisuus RTT ry. Puhelinhaastattelu 31.8.2007. Haastattelija: Reisbacka, Teemu. Haastattelumuistiinpanot haastattelijan hallussa.

[TAK05] Takeuchi, Y. Okamoto, T. Yokoyama, K. Matsuda, S. 2005. A differential analysis approach for improving SOAP processing performance. e-Technology, e-Commerce and e-Service, 2005. EEE '05. Proceedings. Sivut: 472 - 479. INSPEC Accession Number: 8538653

[TID01] Tidwell, Doug. 2001. Programming Web Services with SOAP. Kustantaja: O'Reilly. ISBN: 0-596-00095-2.

[VAL94] Valtioneuvoston päätös rakennustyön turvallisuudesta 23.6.1994/629. [www-dokumentti] Viitattu: 28.1.2008. Saatavilla: http://www.finlex.fi/fi/laki/ajantasa/1994/19940629

[VIL07] Vilkko, Teemu. 2007. Mobiiliteknologia liikkuvan työntekijän apuna. Diplomityö, Lappeenrannan teknillinen yliopisto.

[W3C03] Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, W3C Working Draft 10 November 2003. [www-dokumentti] Viitattu: 17.6.2007. Saatavilla: http://www.w3.org/TR/2003/WD-wsdl20-20031110/

[W3C04] Web Services Architecture, W3C Working Group Note 11 February 2004. www-dokumentti. Viitattu: 17.6.2007. Saatavilla: http://www.w3.org/TR/ws-arch/

[WEI06] Wei, Jun. Hua, Lei. Niu, Chunlei. 2006. Speed-up SOAP processing by data mapping template. Proceedings of the 2006 international workshop on Service-oriented software engineering. Sivut 40 – 46. ISBN:1-59593-398-0

[ZEN07] Zend Studio-kotisivut. 2007. [www-dokumentti] Viitattu: 29.11.2007 Saatavilla: http://www.zend.com/en/products/studio/features

[ZHA02] Davis, Alexander. Zhang, Du. A Compararative Study of DCOM and SOAP. 2002. Multimedia Software Engineering, Joulukuu / 2002. Proceedings. Fourth International Symposium on. Sivut: 48 – 55. Digital Object Identifier 10.1109/MMSE.2002.1181595

Liite 1. getErrors-funktion SOAP-viestit

```
POST /webservice/mobile_service.php HTTP/1.1
Host: localhost
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length
```

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="urn:MobildingServer" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
<ns1:getErrors>
  <rfid xsi:type="xsd:string">E0040100074A3CFE</rfid>
  <login xsi:type="xsd:string">test</login>
  <password xsi:type="xsd:string">test</password>
</ns1:getErrors>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```
HTTP/1.1 200 OK
Content-Length: 10410
Connection: Keep-Alive
Content-Type: text/xml; charset=UTF-8
```

```
<?xml version="1.0" encoding="UTF-8"?>

<SOAP-ENV:Envelope  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:ns4="urn:MobildingServer"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>

<ns4:getErrorsResponse>
<success xsi:type="xsd:boolean">true</success>
<errors xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="ns4:errors[1]" SOAP-
ENC:offset="[0]">
  <item>
  <error_type xsi:type="xsd:string">2</error_type>
  <error_desc xsi:type="xsd:string">Elementin mittauksessa havaittiin virhe ja mitatut tulokset
eroavat liikaa      suunnitelluista mitoista.</error_desc>
  <error_id xsi:type="xsd:string">67</error_id>
  </item>
</errors>
</ns4:getErrorsResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Liite 2. Mobile Service WSDL-määritys

```
<?xml version="1.0" encoding="utf-8"?>
<definitions name="MobildingServer" targetNamespace="urn:MobildingServer"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="urn:MobildingServer"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns="http://schemas.xmlsoap.org/wsdl/">
    <types xmlns="http://schemas.xmlsoap.org/wsdl/">
        <schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:MobildingServer">
            <complexType name="errorlist">
                <complexContent>
                    <restriction base="SOAP-ENC:Array">
                        <attribute ref="SOAP-ENC:arrayType" wsdl:arrayType="xsd:int[]" />
                    </restriction>
                </complexContent>
            </complexType>
            <complexType name="errors">
                <all>
                    <element name="error_type" type="xsd:int" />
                    <element name="error_desc" type="xsd:string" />
                    <element name="error_id" type="xsd:int" />
                </all>
            </complexType>
            <complexType name="errorsArray">
                <complexContent>
                    <restriction base="SOAP-ENC:Array">
                        <attribute ref="SOAP-ENC:arrayType" wsdl:arrayType="tns:errors[]"
/>
                    </restriction>
                </complexContent>
            </complexType>
            <complexType name="storage">
                <all>
                    <element name="gpslat" type="xsd:double" />
                    <element name="gpslong" type="xsd:double" />
                    <element name="storageid" type="xsd:string" />
                    <element name="rfid" type="xsd:string" />
                </all>
            </complexType>
            <complexType name="storageArray">
                <complexContent>
                    <restriction base="SOAP-ENC:Array">
                        <attribute ref="SOAP-ENC:arrayType" wsdl:arrayType="tns:storage[]"
/>
                    </restriction>
                </complexContent>
            </complexType>
            <complexType name="elementinfo">
                <all>
                    <element name="elementid" type="xsd:string" />
                    <element name="shape" type="xsd:string" />
                    <element name="width" type="xsd:int" />
                    <element name="height" type="xsd:int" />
                    <element name="thickness" type="xsd:int" />
                    <element name="weight" type="xsd:int" />
                    <element name="width_measured" type="xsd:int" />
                    <element name="height_measured" type="xsd:int" />
                    <element name="thickness_measured" type="xsd:int" />
                    <element name="weight_measured" type="xsd:int" />
                    <element name="cross_measured" type="xsd:int" />
                    <element name="planned_installdate" type="xsd:int" />
                    <element name="currentstate" type="xsd:int" />
                    <element name="reported_errors" type="tns:errorlist" />
                </all>
            </complexType>
```

(jatkuu)

```
    </schema>
</types>
<message name="sendLocationRequest">
    <part name="rfid" type="xsd:string" />
    <part name="gpslat" type="xsd:double" />
    <part name="gpslong" type="xsd:double" />
    <part name="storageid" type="xsd:string" />
    <part name="login" type="xsd:string" />
    <part name="password" type="xsd:string" />
</message>
<message name="sendLocationResponse">
    <part name="message" type="xsd:string" />
    <part name="success" type="xsd:boolean" />
</message>
<message name="getInfoRequest">
    <part name="rfid" type="xsd:string" />
    <part name="login" type="xsd:string" />
    <part name="password" type="xsd:string" />
</message>
<message name="getInfoResponse">
    <part name="return" type="tns:elementinfo" />
</message>
<message name="sendStateRequest">
    <part name="rfid" type="xsd:string" />
    <part name="newstate" type="xsd:int" />
    <part name="state_time" type="xsd:int" />
    <part name="login" type="xsd:string" />
    <part name="password" type="xsd:string" />
</message>
<message name="sendStateResponse">
    <part name="message" type="xsd:string" />
    <part name="success" type="xsd:boolean" />
</message>
<message name="reportErrorRequest">
    <part name="rfid" type="xsd:string" />
    <part name="error_type" type="xsd:int" />
    <part name="error_desc" type="xsd:string" />
    <part name="login" type="xsd:string" />
    <part name="password" type="xsd:string" />
</message>
<message name="reportErrorResponse">
    <part name="success" type="xsd:boolean" />
    <part name="message" type="xsd:string" />
    <part name="error_id" type="xsd:int" />
</message>
<message name="sendErrorFileRequest">
    <part name="error_id" type="xsd:int" />
    <part name="data" type="xsd:string" />
    <part name="file_extension" type="xsd:string" />
    <part name="login" type="xsd:string" />
    <part name="password" type="xsd:string" />
</message>
<message name="sendErrorFileResponse">
    <part name="success" type="xsd:boolean" />
    <part name="error" type="xsd:string" />
</message>
<message name="updateErrorStateRequest">
    <part name="error_id" type="xsd:int" />
    <part name="error_state" type="xsd:int" />
    <part name="state_message" type="xsd:string" />
    <part name="login" type="xsd:string" />
    <part name="password" type="xsd:string" />
</message>
<message name="updateErrorStateResponse">
    <part name="success" type="xsd:boolean" />
    <part name="error" type="xsd:string" />
</message>
<message name="getStoragesRequest">
    <part name="shape" type="xsd:string" />
    <part name="login" type="xsd:string" />
    <part name="password" type="xsd:string" />
</message>
```

(jatkuu)

(liite 2 jatkoa)

```
<message name="getStoragesResponse">
    <part name="success" type="xsd:boolean" />
    <part name="storage" type="tns:storageArray" />
</message>
<message name="sendMeasurementsRequest">
    <part name="rfid" type="xsd:string" />
    <part name="width" type="xsd:int" />
    <part name="height" type="xsd:int" />
    <part name="thickness" type="xsd:int" />
    <part name="cross" type="xsd:int" />
    <part name="login" type="xsd:string" />
    <part name="password" type="xsd:string" />
</message>
<message name="sendMeasurementsResponse">
    <part name="success" type="xsd:int" />
    <part name="error" type="xsd:string" />
</message>
<message name="getErrorsRequest">
    <part name="rfid" type="xsd:string" />
    <part name="login" type="xsd:string" />
    <part name="password" type="xsd:string" />
</message>
<message name="getErrorsResponse">
    <part name="success" type="xsd:boolean" />
    <part name="errors" type="tns:errorsArray" />
</message>
<message name="getMapLocationRequest">
    <part name="dim_x" type="xsd:int" />
    <part name="dim_y" type="xsd:int" />
    <part name="storagearea" type="xsd:string" />
    <part name="shape" type="xsd:string" />
    <part name="scale" type="xsd:int" />
    <part name="draw_storage_area" type="xsd:boolean" />
    <part name="login" type="xsd:string" />
    <part name="password" type="xsd:string" />
</message>
<message name="getMapLocationResponse">
    <part name="success" type="xsd:boolean" />
    <part name="message" type="xsd:string" />
    <part name="filetype" type="xsd:string" />
    <part name="file" type="xsd:string" />
    <part name="coord_ltop_x" type="xsd:double" />
    <part name="coord_ltop_y" type="xsd:double" />
    <part name="coord_rbottom_x" type="xsd:double" />
    <part name="coord_rbottom_y" type="xsd:double" />
</message>
<portType name="MobildingServerPort">
    <operation name="sendLocation">
        <input message="tns:sendLocationRequest" />
        <output message="tns:sendLocationResponse" />
    </operation>
    <operation name="getInfo">
        <input message="tns:getInfoRequest" />
        <output message="tns:getInfoResponse" />
    </operation>
    <operation name="sendState">
        <input message="tns:sendStateRequest" />
        <output message="tns:sendStateResponse" />
    </operation>
    <operation name="reportError">
        <input message="tns:reportErrorRequest" />
        <output message="tns:reportErrorResponse" />
    </operation>
    <operation name="sendErrorFile">
        <input message="tns:sendErrorFileRequest" />
        <output message="tns:sendErrorFileResponse" />
    </operation>
    <operation name="updateErrorState">
        <input message="tns:updateErrorStateRequest" />
        <output message="tns:updateErrorStateResponse" />
    </operation>
```

(jatkuu)

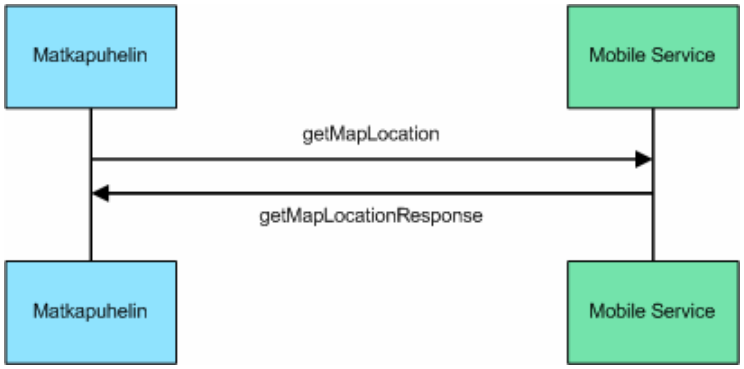
```
        <operation name="getStorages">
            <input message="tns:getStoragesRequest" />
            <output message="tns:getStoragesResponse" />
        </operation>
        <operation name="sendMeasurements">
            <input message="tns:sendMeasurementsRequest" />
            <output message="tns:sendMeasurementsResponse" />
        </operation>
        <operation name="getErrors">
            <input message="tns:getErrorsRequest" />
            <output message="tns:getErrorsResponse" />
        </operation>
        <operation name="getMapLocation">
            <input message="tns:getMapLocationRequest" />
            <output message="tns:getMapLocationResponse" />
        </operation>
    </portType>
    <binding name="MobildingServerBinding" type="tns:MobildingServerPort">
        <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
        <operation name="sendLocation">
            <soap:operation soapAction="urn:MobildingServer#MobildingServer#sendLocation"
/>
            <input>
                <soap:body use="encoded" namespace="urn:MobildingServer"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
            </input>
            <output>
                <soap:body use="encoded" namespace="urn:MobildingServer"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
            </output>
        </operation>
        <operation name="getInfo">
            <soap:operation soapAction="urn:MobildingServer#MobildingServer#getInfo" />
            <input>
                <soap:body use="encoded" namespace="urn:MobildingServer"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
            </input>
            <output>
                <soap:body use="encoded" namespace="urn:MobildingServer"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
            </output>
        </operation>
        <operation name="sendState">
            <soap:operation soapAction="urn:MobildingServer#MobildingServer#sendState" />
            <input>
                <soap:body use="encoded" namespace="urn:MobildingServer"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
            </input>
            <output>
                <soap:body use="encoded" namespace="urn:MobildingServer"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
            </output>
        </operation>
        <operation name="reportError">
            <soap:operation soapAction="urn:MobildingServer#MobildingServer#reportError" />
            <input>
                <soap:body use="encoded" namespace="urn:MobildingServer"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
            </input>
            <output>
                <soap:body use="encoded" namespace="urn:MobildingServer"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
            </output>
        </operation>
        <operation name="sendErrorFile">
            <soap:operation soapAction="urn:MobildingServer#MobildingServer#sendErrorFile"
/>
            <input>
                <soap:body use="encoded" namespace="urn:MobildingServer"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
            </input>
```

(jatkuu)

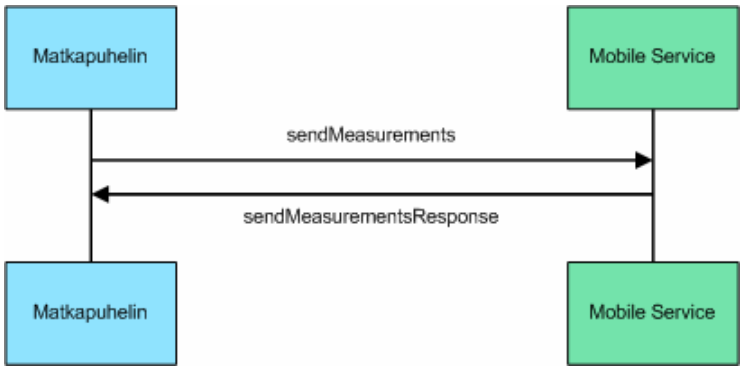
```
            <output>
                <soap:body use="encoded" namespace="urn:MobildingServer"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
            </output>
        </operation>
        <operation name="updateErrorState">
            <soap:operation
soapAction="urn:MobildingServer#MobildingServer#updateErrorState" />
            <input>
                <soap:body use="encoded" namespace="urn:MobildingServer"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
            </input>
            <output>
                <soap:body use="encoded" namespace="urn:MobildingServer"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
            </output>
        </operation>
        <operation name="getStorages">
            <soap:operation soapAction="urn:MobildingServer#MobildingServer#getStorages" />
            <input>
                <soap:body use="encoded" namespace="urn:MobildingServer"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
            </input>
            <output>
                <soap:body use="encoded" namespace="urn:MobildingServer"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
            </output>
        </operation>
        <operation name="sendMeasurements">
            <soap:operation
soapAction="urn:MobildingServer#MobildingServer#sendMeasurements" />
            <input>
                <soap:body use="encoded" namespace="urn:MobildingServer"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
            </input>
            <output>
                <soap:body use="encoded" namespace="urn:MobildingServer"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
            </output>
        </operation>
        <operation name="getErrors">
            <soap:operation soapAction="urn:MobildingServer#MobildingServer#getErrors" />
            <input>
                <soap:body use="encoded" namespace="urn:MobildingServer"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
            </input>
            <output>
                <soap:body use="encoded" namespace="urn:MobildingServer"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
            </output>
        </operation>
        <operation name="getMapLocation">
            <soap:operation soapAction="urn:MobildingServer#MobildingServer#getMapLocation"
/>
            <input>
                <soap:body use="encoded" namespace="urn:MobildingServer"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
            </input>
            <output>
                <soap:body use="encoded" namespace="urn:MobildingServer"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
            </output>
        </operation>
    </binding>
    <service name="MobildingServerService">
        <documentation />
        <port name="MobildingServerPort" binding="tns:MobildingServerBinding">
            <soap:address location="http://157.24.188.71/webservice/mobile_service.php" />
        </port>
    </service>
</definitions>
```

Liite 3. Viestien MSC-kuvaukset

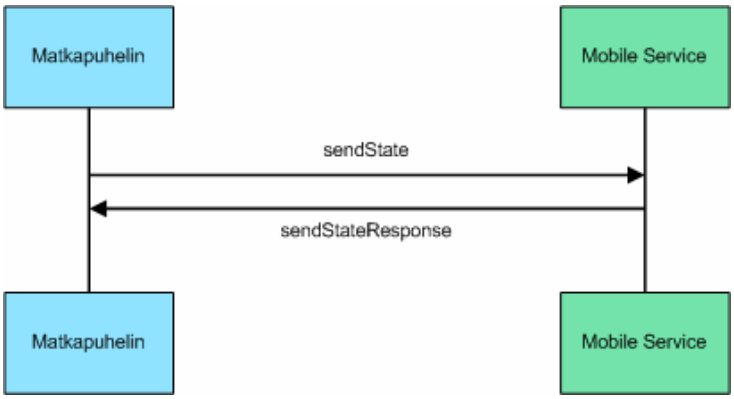
Kartan hakeminen: getMapLocation



Mittatietojen lähettäminen: sendMeasurements

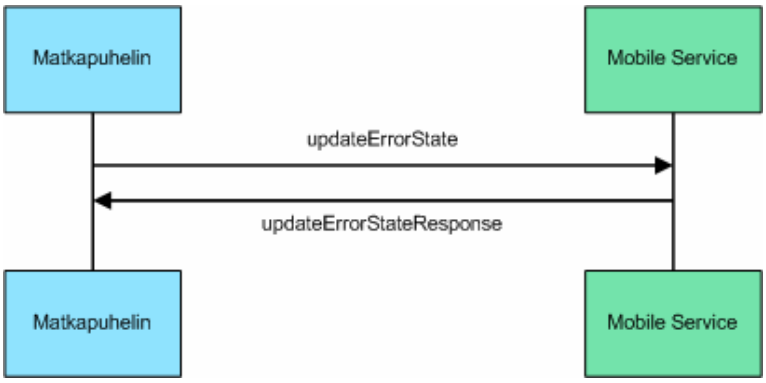


Tilan päivittäminen: sendState

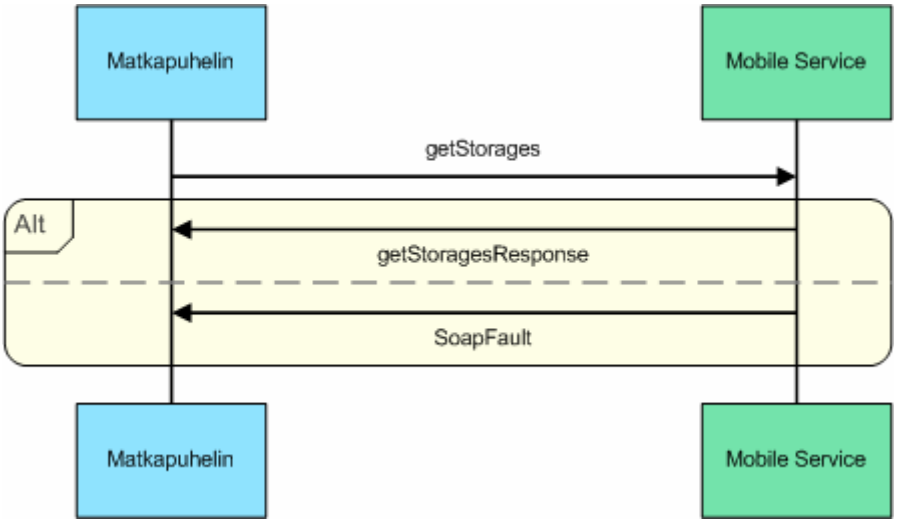


(jatkuu)

Virhetilan päivittäminen: updateErrorState



Varastojen haku: getStorages



Liite 4. Elementtien tilat järjestelmässä

Mobilding-tilan tunnus	Tila	Kuvaus
1	Not started	Ei aloitettu
2	Scheduled	Odottaa valmistusta
3	Cut	Leikattu
4	In Assembly	Kootaan
5	In Galvanizer	Galvanoijassa
6	Painted	Maalattu
7	Completed, In Yard	Valmis, tehtaalla
8	Completed, Shipped	Valmis, lähetetty
9	In transit	Kuljetuksessa
10	On Site	Rakennustyömaalla
11	Erected	Asennettu
12	Scheduled Fabrication Date	Suunniteltu valmistus
13	Date Issued To Shop	Tilattu tehtaalta
14	Planned Erection Date	Suunniteltu asennus