LAPPEENRANNAN TEKNILLINEN YLIOPISTO
TIETOTEKNIIKAN OSASTO

PAIKKA- JA TILANNETIEDON HYÖDYNTÄMINEN SOVELLUKSISSA

Diplomityön aihe on hyväksytty Tietotekniikan
osaston osastoneuvostossa 17.08.2005
Työn tarkastajat: Jouni Ikonen, Kari Heikkinen
Työn ohjaaja: Kari Heikkinen

Kimmo Koskinen
Orioninkatu 1 A 7
53850 Lappeenranta
kimmo.m.koskinen@lut.fi
050-5338 473

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto

Tietotekniikan osasto

Kimmo Koskinen

Paikka- ja tilannetiedon hyödyntäminen sovelluksissa

Diplomityö

2005

56 sivua, 13 kaaviota, 4 taulukkoa ja 4 liitettä.

Tarkastajat: Dosentti Jouni Ikonen, TkT Kari Heikkinen

Hakusanat: paikkatieto, tilannetieto

Erilaisten mobiiliverkkojen käytön yleistyessä nousee esiin uudenlaisia sovellusalueita, kuten esimerkiksi paikkatietoiset sovellukset. Mobiiliudesta johtuen sovellusten käyttötilanteet vaihtelevat. Käyttötilanteista voidaan kerätä tietoa ja käyttää tätä hyödyksi. Tilannetiedolla tarkoitetaan sovelluksen käyttötilanteeseen tai käyttäjään liittyvää lisätietoa.

Paikka- ja tilannetietoisten sovellusten kehittäminen vaati monia ohjelmistokehitystä tukevia järjestelmiä. Tilannetiedon väljän määritelmän takia tilannetietoisten sovellusten kehitykselle ei ole vielä selkeitä toimintamalleja. Tilannetietoisten sovellusten kehitystä avustavia järjestelmiä on luotu etenkin tutkimuksessa, mutta nämä eivät ole vielä yleistyneet laajempaan käyttöön. Paikkatiedon käyttö sen sijaan on hyvinkin standardoitua, mutta paikkatieto nähdään vain osana tilannetietoa.

Tässä diplomityössä toteutettiin paikka- ja tilannetiedon sovelluskehitystä tukevia järjestelmiä, joilla paikka- ja tilannetiedon hyödyntäminen sovelluksissa mahdollistettiin. WLAN -verkosta saadun paikkatiedon hyödyntämiseen toteutettiin SOAP -palvelurajapinta. Tilannetiedon hyödyntämiseksi toteutettiin MUPE -sovellusympäristöön välittäjäkomponentteja paikka-, sää- ja kuntopyörän harjoitustiedolle sekä RFID -havaintotiedoille. Näitä komponetteja käytettiin tilannetietoisten sovellusten luomiseen sekä tietoliikennetekniikan laitoksen codecamp -kursseilla, että tilannetietoisessa pelisovelluksessa. Työn tuloksena saatiin toimivia sovelluksia, ja välittäjäkomponentit sovellusten luomiseen. Työn tuloksena voidaan todeta, että ilman tilannetietoista sovelluskehitystä tukevia komponentteja, olisi tämäntyyppinen sovelluskehitys huomattavasti vaativampaa. Tukevat komponentit helpottavat sovelluskehitystä, mutta helposti myös rajaavat kehitysmahdollisuuksia.

ABSTRACT

Lappeenranta University of Technology

Department of Information Technology

Kimmo Koskinen

Utilisation of Location and Context Information in Applications

Masters Thesis

2005

56 pages, 13 figures, 4 tables and 4 appendices.

Supervisors: Docent Jouni Ikonen, Dr. Tech. Kari Heikkinen

Keywords: location information, context information

New application areas, such as location aware applications, are emerging with the development of different mobile networks. Due to mobility, the context in which applications are used varies. Context information can be gathered and used in applications. Context information means in this thesis information related to the user or the usage of the application.

Supporting systems are needed for the development of location and context aware software. Due to the loose definition of context information, there are no distinguished functional frameworks for developing context aware applications. System supporting context awareness have been created in research but these are not in common use. Usage of location information is more standardised but location information is usually seen only as part of context information.

In this thesis, systems for supporting context aware software development were implemented and used in software development. A SOAP -interface was implemented to use WLAN cell positioning data. Context information relaying components were implemented for MUPE -platform to provide location, weather, fitness bicycle exercise data and RFID proximity data. These software components were used in creating context aware applications in the courses of the laboratory of communications engineering and in a context aware game application. The results of the work were functional applications and relaying components for use in MUPE application development. As a result of the work it can be concluded that context aware software development would be much more demanding without software components that ease the development process. Supporting components generalise the development process but they can also restrict development possibilities.

ALKUSANAT

Tämä työ on tehty Lappeenrannan teknillisen yliopiston tietoliikennetekniikan laitoksella sekä WLPR.NET että AMPERS -projekteille.

Haluan kiittää työni tarkastajaa Jouni Ikosta hänen luomastaan hyvästä työilmapiiristä tietoliikennetekniikan laitoksella. Hänen avullansa sain kokemusta, jonka saavuttaminen on minulle ainutlaatuista. Haluan kiittää myös ohjaajaani Kari Heikkistä hänen monista ideoistaan sekä uusiin työalueisiin tutustuttamisesta.

Haluan kiittää myös tietoliikennetekniikan laitoksen monia työntekijöitä, Tomi Lapinlampea, Harri Hämäläistä, Radek Spáčilia, Aki Honkasuota, Arto Hämäläistä sekä monia muita, joiden nimen unohdin mainita. Omalta osaltaan autoitte ja rohkaisitte työskentelyäni tietoliikennetekniikan laitoksella ja annoitte monta inspiraatiota.

Erityisesti haluan kiittää Katriina Saransaarta hänen tuestaan työtä kirjoittaessani. Ilman häntä en olisi tähän pisteeseen päässyt.

Sisällysluettelo

Lyhenneluettelo

AMPERS	AMbient and PERsonalized Society
ARP	Address Resolution Protocol
CEP	Context Exchange Protocol
DGPS	Differential GPS
EOTD	Enhanced Observed Time Difference
EPE	Ekahau Positioning Engine
EPSG	European Petroleum Survey Group
GIS	Geographic Information System
GLONASS	GLObal NAvigation Satellite System
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Global System for Mobile Communications
HTML	Hypertext Markup Language
HTTP	HypertTex Transfer Protocol
IDL	Interface Description Language
IE 05	Intelligent Environments 2005
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
J2ME	Java 2 Micro Edition

KKJ ..... Kartastokoordinaattijärjestelmä

LIF ..... Location Interoperability Forum

LIS ..... Location Information System

LMU ..... Location Measurement Unit

LOC ..... Location Working Group

LPP ..... Local Positioning Profile

MAC ..... Media Access Control

MIDP ..... Mobile Information Device Profile

MLP ..... Mobile Location Protocol

MUD ..... Multi User Dungeon

MUPE ..... Multi User Publishing Environment

NMEA ..... National Marine Electronics Association

OGC ..... Open Geospatial Consortium

OMA ..... Open Mobile Alliance

PHP ..... PHP Hypertext Preprocessor

QZSS ..... Quasi-Zenith Satellite System

RADIUS ..... Remote Authentication Dial In User Service

RFID ..... Radio Frequency Identification

RKT-GPS .... Real Time Kinematics GPS

RMI ..... Remote Method Invocation

RSSI ..... Received Signal Strength Intensity

SOAP ..... Simple Object Access Protocol

SQL ..... Structured Query Language

TCP ..... Transmission Control Protocol

UML ..... Unified Modelling Language

WAP ..... Wireless Application Protocol

WAWC04 ..... 2nd Workshop on Applications of Wireless Communications

WGS 84 ..... World Geodetic System 84

WLAN ..... Wireless Local Area Network

WSDL ..... Web Services Description Language

WWW ..... World Wide Web

XML ..... Extensible Markup Language

Kuvat

Taulukot

Luku 1

Johdanto

Erilaisten mobiiliverkkojen käytön yleistyessä nousee esiin uudenlaisia sovellusalueita. Tällaisia sovellusalueita ovat esimerkiksi paikkatietoiset sovellukset. Mobiiliudesta johtuen sovellusten käyttötilanteet vaihtelevat. Eri käyttötilanteista voidaan myös kerätä tietoa ja käyttää tätä hyödyksi sovelluksissa. Käyttäjän käyttötilanteeseen liittyvää tietoa voi olla esimerkiksi paikallisen sään lämpötila. Tilannetiedolla (konteksti- tai asiayhteystieto) tarkoitetaan sovelluksen käyttötilanteeseen tai käyttäjään liittyvää lisätietoa. Paikkatieto nähdään usein osana tilannetietoa. Esimerkki tilannetietoisesta sovelluksesta on vaikkapa kaupunkiopassovellus matkapuhelimessa, joka nähtävyyden ohi käveltäessä muistuttaa läheisen ravintolan lounastarjouksesta.

Paikkatiedon keräämiseen on kehitetty erilaisia tekniikoita. Näitä ovat mm. satelliittipaikannus ja maanpäällisten tietoliikenneverkkojen käyttäminen käyttäjien päätelaitteiden paikan määrittämisessä. Riippuen käytetystä paikannusjärjestelmästä, vaihtelee paikkatiedon saantitapa, tarkkuus, varmuus sekä kattavuus. Tässä työssä tarkastellaan erilaisia paikantamiseen käytettävissä olevia järjestelmiä, sekä esitellään WLAN -verkkoon kehitetty paikannusrajapinta.

Tilannetiedolla voidaan käsittää hyvinkin monia asioita. Paikkatiedon määrittelemä sijaintitieto voidaan nähdä yhtenä tilannetiedon osana. Esimerkiksi käyttäjän kulloinenkin sijainti on osa käyttäjään liittyvää tilannetietoa. Samoin käyttäjän sijainnissa vallitseva lämpötila on esimerkki tilannetiedosta. *Eri lähteistä koostuvan tilannetiedon hyödyntäminen sovelluksissa vaatii ymmärrystä eri tietojärjestelmistä (esimerkiksi tietotyypit, rajapinnat, protokollat).* Tilannetiedon lähteet voivat olla hyvinkin vaihtelevia riippuen tilannetiedon tyypistä. Eri lähteiden käyttöä taas helpottavat ohjelmistoalustat, joilla tilannetiedon

käyttöä voidaan hallita. Tällaisen käytön hallintaan kuuluu mm. yhtenäiset protokollat ja välittäjäohjelmistot (esimerkiksi MUPE (Multi User Publishing Environment)[1]). Kun tilannetieto on saatu sovellukseen, voidaan sillä laukaista toimintoja esimerkiksi ennalta määritellyin säännöin. Esimerkiksi MUPE -ympäristö mahdollistaa edellämainituin tavoin nopean sovelluskehityksen MIDP (Mobile Information Device Profile) -laiteympäristössä. Erityisyys tällä alustalla on tilannetiedon huomioonotto.

Diplomityön tavoitteena on analysoida paikka- ja tilannetiedon keruujärjestelmiä, välitysjärjestelmiä (rajapinnat) sekä hyödyntämistä sovelluksissa. Tavoitteet pyritään todentamaan sovelluksien avulla. Työn arviointiosuudessa käydään läpi paikka- ja tilannetietoisten ohjelmistojen kehityksessä esiin tulleita ongelmia käytettyjen ohjelmistoalustojen ja metodien avulla.

Diplomityö on tehty LTY:n (Lappeenrannan Teknillisen Yliopiston) tutkimuskeskus TBRC:n alaisessa AMPERS (AMbient and PERsonalized Society) -projektissa, jossa kehitettiin sovelluksia MUPE -ympäristön avulla. Osa työstä on tehty LTY:n Tietoliikennetekniikan Osastolla WLPR.NET -projektissa. Paikkatiedon keräämiseen on käytetty WLPR.NET -projektissa luotua WLAN (Wireless Local Area Network) -verkkoa.

Tämä diplomityö koostuu viidestä kappaleesta. Ensimmäisessä kappaleessa (johdanto) esitetään ratkaistava ongelma ja määritellään ongelma-alue. Tämän lisäksi esitetään tavoitteet ja menetelmät joilla ongelma pyritään ratkaisemaan. Toisessa kappaleessa analysoidaan paikka- ja tilannetiedon nykytila. Kolmannessa kappaleessa todennetaan tehdyt ratkaisut sovelluksien avulla. Neljännessä kappaleessa arvioidaan toteutettuja ratkaisuja. Viidennessä kappaleessa esitetään johtopäätökset.

[1] `http://www.mupe.net`

Luku 2

Paikka- ja tilannetieto

Erilaisten mobiiliverkkojen yleistymisen johdosta tietoteknisiä sovelluksia käytetään usein paikasta riippumatta eri käyttötilanteissa. Päätelaitteiden sijainti pystytään olemassaolevin keinoin päättelemään mahdollisesti usean eri järjestelmän avulla. Paikkatiedon sisältö ja tarkkuus riippuu paikkatiedon hankintaan käytetystä järjestelmästä. Seuraavassa kappaleessa tarkastellaan paikkatiedon olemuksen määrittelyjä sekä erilaisia järjestelmiä, joilla paikkatietoa voidaan kerätä.

Tilannetiedolla taas käsitetään yleisesti erilaisiin käyttötilanteisiin liittyvää tietoa. Tämä tieto on yleisesti paljon monimuotoisempaa kuin paikkatieto. Tämä näkyy myös erilaisten tilannetietoisten sovellusten laajuudessa. Yleensä tilannetiedon käsitetään olevan kuitenkin samassa asemassa paikkatiedon kanssa; sitä voidaan käyttää palvelujen lisäarvona.

Paikka- ja tilannetieto voidaan yleensä liittää johonkin käyttäjään, joka on luonnollinen henkilö. Monesti voi tulla tilanteita, jolloin oman paikkatiedon antaminen muille ei ole haluttavaa, koska se saattaisi loukata yksityisyyttä. Koska paikka- sekä tilannetieto ovat käyttäjän henkilökohtaista tietoa, on otettava huomioon henkilökohtaisen tiedon käsittelystä säädetyt lait ja määräykset. Paikka- ja tilannetietojen yksityisyyttä on käsitelty myös tieteellisessä tutkimuksessa. Esimerkiksi *The Active Badge Location System* -järjestelmässä [1] tutkittiin käyttäjien paikantamista ja paikkatiedon hyväksikäyttöä infrapunalähettimien avulla. Tutkimuksessa rakennettiin infrapunalähettimen sisältävien rintamerkkien seurantajärjestelmä, jota käytettiin hyväksi mm. puhelinvaihteen työn helpottamiseksi työntekijöiden paikantamisessa. Tutkimuksen aikana huomattiin, että vaikka paikannusjärjestelmä olikin toiminnaltaan hyvä, eivät työntekijät olleet vakuuttuneita henkilökohtaisen tietosuojan hoitamisesta. Päähuomio oli, että järjestelmää pitää koekäyttää, että käytön seuraukset

tulisivat jokaiselle selväksi. Tutkimuksessa huomioitiin myös, että yleinen käyttömuoto oli jättää infrapunalähetin työpöydälle, kun oikeaa sijaintitietoa ei haluttu välittää. Onkin huomattava että paikannusjärjestelmän paikantaman laitteen ja käyttäjän välinen yhteys on häilyvä.

Suomen lainsäädännössä paikkatiedosta on säädetty 16.6.2004 voimaan tulleessa Sähköisen viestinnän tietosuojalaissa [2]. Tätä ennen paikkatietoa erityisesti käsittelevää lakia ole suomessa ollut. Aiemmin paikkatietoa koskevia määrityksiä on jouduttu tulkitsemaan sekä henkilösuojalain että lain yksityisyyden suojasta televiestinnässä ja teletoiminnan tietoturvasta mukaan. Sähköisen viestinnän tietosuojalaissa määritellään paikkatiedon olevan:

> "tietoa, joka ilmaisee liittymän tai päätelaitteen maantieteellisen sijainnin ja jota käytetään muuhun tarkoitukseen kuin verkkopalvelun tai viestintäpalvelun toteuttamiseen"

Paikkatiedoksi määritellään siis erityisesti paikannuspalveluissa käytettävä käyttäjän paikkatieto. Pääpiirteet paikkatiedon käytöstä on käsitelty luvussa 4. Lain mukaan ketään ei saa paikantaa ilman suostumusta. Paikannukseen on siis saatava palvelukohtainen suostumus ennen kuin paikannus voidaan aloittaa. Alle 15 -vuotiaan paikannuspalveluiden käytöstä päättää huoltaja. Käyttäjän on myös pystyttävä saamaan tietoa paikkatietojen tarkkuudesta, käytön tarkoituksesta, kestosta sekä mahdollisuudesta luovuttaa paikkatietoja kolmannelle osapuolelle. Käyttäjälle tulee olla myös helppo ja maksuton tapa kieltää paikkatietojen käsittely. Hätätilanteesssa on paikannuksen kuitenkin oltava aina mahdollista.

Hätäpaikannuksesta on säädetty myös yhdysvalloissa E911 -säännöksellä[1]. Tämän mukaan hätätilanteessa operaattorin tulee pystyä antamaan soiton tehneen puhelimen sijainti. Euroopan laajuisesti vastaavaa on säädetty Euroopan komission E112 -säädöksessä. Hätätilanteissa paikkatiedon käyttö on hyvinkin oikeutettua. Yleisesti voidaan sanoa, että paikkatiedon käsittelyssä tulisi noudattaa eettisiä sääntöjä, jotta käyttäjien yksityisyyttä ei loukattaisi.

2.1 Paikkatieto

Jotta paikkatietoisia sovelluksia voitaisiin rakentaa, tarvitaan paikkatiedolle määrittely. Paikkatiedon määrittely voi olla hyvinkin sovelluskohtaista, riippuen sovelluksen käyttöalueesta, paikkatiedon tarkkuudesta ja paikannettavista kohteista. Paikkatiedon määrittelyssä

[1] `http://www.fcc.gov/911/enhanced/`

tulisi erottaa paikkatietoa tuottavat järjestelmät ja paikkatietoa kuvaavat tietorakenteet.

Paikkatiedolle on olemassa useita eri määrityksiä riippuen käyttöyhteydestä. Esimerkiksi Sanastokeskus (TSK, Tekniikan Sanastokeskus) on koostanut paikkatiedosta määrityksen Paikannussanasto -julkaisussa [3]. Paikkatieto on tämän julkaisun mukaan määritelty seuraavasti:

> "*paikannettua* kohdetta kuvaavan sijaintitiedon (1) ja kohteen ominaisuuksia kuvaavien tietojen muodostama kokonaisuus"

Sijaintitiedolla (engl. location information) tarkoitetaan jonkin kohteen sijaintia vertausjärjestelmässä (engl. reference system). Vertausjärjestelmän avulla voidaan ilmaista yksiselitteisesti jonkin kohteen sijainti. Vertausjärjestelmiä ovat mm. erilaiset koordinaattijärjestelmät, kuten Suomessa käytössä oleva KKJ (Kartastokoordinaattijärjestelmä) [4], mutta myös erilaiset katuosoitejärjestelmät.

Kohteen ominaisuuksia voivat olla vaikkapa väri tai kohteen koko. Ominaisuustiedon liittäminen voi kuitenkin olla hyvin sovelluskohtaista. Eri sovellukset hyödyntävät yleensä erilaisia ominaisuustietoja ja kaikkia ominaisuustietoja ei voida olettaa järkeväksi käsitellä kaikissa sovelluksissa.

Sijaintitiedon ilmaisemisen tarkkuuteen liittyvät sekä käytetty vertausjärjestelmä että järjestelmä, jolla paikkatietoa kerätään. Maailmanlaajuisesti katsottuna eri mailla on perinteisesti ollut omat vertausjärjestelmänsä. Lisäksi maiden sisällä eri kaupungeilla voi myös olla käytössä omat vertausjärjestelmänsä. Koordinaattipohjaiset vertausjärjestelmät pohjaavat tiettyyn geodeettiseen datumiin. Geodeettisillä datumeilla määritetään maapallon koko ja muoto sekä käytetyn vertausjärjestelmän origo sekä orientaatio [5]. Eri vertausjärjestelmät voivat pohjata eri datumeihin, joka on otettava huomioon tehtäessä muunnoksia koordinaatistosta toiseen. EPSG (European Petroleum Survey Group) pitää kirjaa yleisimmistä koordinaattivertausjärjestelmistä sekä parametreista, joita tarvitaan muunnoksiin eri koordinaatistojen välillä[2]. Muunnoksista voi aiheutua virheitä riippuen tehdäänkö muunnos kahden eri datumia käyttävän koordinaattivertausjärjestelmän välillä.

Vertausjärjestelmien moninaisuudesta johtuen yhteiselle vertausjärjestelmälle on ollut tarve jotta paikkatietoa voitaisiin käyttää maailmanlaajuisesti. WGS 84 (World Geodetic System 84) [6] on maailmanlaajuinen vertausjärjestelmä, joka on käytössä myös GPS (Global Positioning System) -paikannusjärjestelmässä. Paikkatiedon käytön skaala ei kuitenkaan aina ole maailmanlaajuinen. Esimerkiksi rakennuksien sisätilojen paikkatietoa ei välttä-

[2] http://www.epsg.org/

mättä ole mielekästä esittää maailmanlaajuisella koordinaattijärjestelmällä. Rakennuksien sisällä tarkkuudeksi voi riittää esimerkiksi huonekohtainen paikkatieto, mutta tätä tietoa voi olla vaikea ilmaista käyttämällä esim. WGS 84 -koordinaattijärjestelmää. Huonekohtainen paikkatieto voi olla mielekkäämpää sitoa vaikkapa rakennuksen pohjapiirustuksiin tai huoneiden numerointijärjestelmään.

Sijaintitietoa, joka on esitetty yhdessä vertausjärjestelmässä, voidaan käyttää myös muissa vertausjärjestelmissä, mikäli näiden vertausjärjestelmien välillä voidaan tehdä muunnoksia. Esimerkiksi Suomessa käytetetyn valtakunnallisen kartastokoordinaattijärjestelmän (KKJ) ja maailmanlaajuisen WGS 84 -järjestelmän välillä on mahdollista tehdä muunnos, jolloin paikkatietoa voidaan käsitellä kummassakin järjestelmässä. Tämä voi olla hyödyllistä esimerkiksi silloin kun käytössä on vain jommassa kummassa järjestelmässä esitettyjä karttoja.

Paikkatiedon koostumus vaihtelee paikkatietoa tuottavien järjestelmien välillä. Esimerkiksi solupaikannusjärjestelmä, kuten WLAN -solupaikannus, saattaa tarjota paikkatietona vain WLAN -solun tunnisteen. WLAN -solun kattama alue taas voidaan kuvata jossakin vertausjärjestelmässä. Näin saatu paikkatieto ei ole yksiselitteistä. Myös soluhavaintoon voi liittyä epävarmuutta, esimerkiksi jos havaintohetkestä on jo kulunut pitkä aika.

Epävarmuutta voi syntyä myös edellämainituista koordinaattimuunnoksista. Eri vertausjärjestelmät voivat esittää sijaintitietoa sisäisesti eri tavoin, jolloin koordinaattimuunnokset eivät ole välttämättä yksiselitteisiä.

2.2 Paikkatiedon lähteet

Paikkatiedon lähteinä toimivat erilaiset paikannusta tekevät järjestelmät. Tuotetun paikkatiedon tietotyyppi voi vaihdella lähteenä käytetyn järjestelmän mukaan. Esimerkiksi saatu sijaintitieto voi olla esitetty eri koordinaattijärjestelmissä. Paikkatiedon lähdejärjestelmiä on olemassa monia. Sekä pelkästään paikannuskäyttöön kehitettyjä, että paikkatiedon tuottoon erilaisia olemassaolevia tietojärjestelmiä hyödyntäviä järjestelmiä. Tämän työn aihepiirissä tärkeimpiä järjestelmiä on käyty läpi seuraavissa kappaleissa.

2.2.1 Satelliittipaikannusjärjestelmät

Satelliittipaikannusjärjestelmissä määritetään maapallon pinnalla olevien kohteiden sijainti maapalloa kiertävien satelliittien avulla. Satelliittipaikannusjärjestelmistä tunnetuin lienee

Yhdysvaltojen puolustushallinnon kehittämä GPS -paikannusjärjestelmä. Satelliittipaikannusjärjestelmiä on erilaisia, mutta yleinen määritelmä niille löytyy mm. Teknologian kehittämiskeskuksen (Tekes) julkaisussa Paikannus mobiilipalveluissa ja sovelluksissa [7] (kpl 2.5, s.9):

> "Satelliittipaikannus perustuu paikannussatelliittien lähettämään ratatieto- ja aikamerkkisignaalien vastaanottoon ja vastaanottimen sijainnin laskemiseen satelliittien etäisyyksien perusteella."

GPS -järjestelmän lisäksi Euroopassa on kehitteillään Galileo -niminen satelliittipaikannusjärjestelmä. Galileo -järjestelmän kehittämisen syynä on mm. pyrkimys saada eurooppalaisten käyttöön satelliittipaikannusjärjestelmä, joka ei ole riippuvainen yhdysvaltain GPS järjestelmästä. Yhdysvaltojen lisäksi satelliittipaikannusjärjestelmiä on kehittänyt myös Venäjä, jolla on käytössä GLONASS (GLObal NAvigation Satellite System) -satelliittipaikannusjärjestelmä[3]. GLONASS -järjestelmä ei tosin ole yhtä käytetty kuin GPS -järjestelmä.

Satelliittipaikannusjärjestelmät voivat tarjota hyvinkin tarkkaa sijaintitietoa. Esimerkiksi GPS -järjestelmässä ilmakehän aiheuttamia virheitä satelliittien lähettämän signaalin vastaanotossa voidaan korjata DGPS (Differential GPS) -järjestelmän avulla. DGPS -järjestelmässä GPS -vastaanottimelle välitetään korjaustietoa vastaanottimen läheisyydessä sijaitsevalta korjausasemalta. Korjausasema lähettää oman tarkan sijaintinsa ja GPS:n avulla lasketun sijainnin eron, jota GPS -vastaanottimet voivat käyttää oman sijaintinsa korjaamiseen. DGPS:n avulla voidaan päästä noin 2 metrin tarkkuuteen ([7], s 10). DGPS -järjestelmällä tarkkuuden nostaminen on ollut hyödyllistä etenkin ennen kuin GPS -järjestelmän siviilipaikannuksen tarkkuutta häiritevä "Selective Availability" -ominaisuus[4] poistettiin käytöstä. DGPS -järjestelmässä vastaanottimien tulee olla korjausdataa lähettävien asemien läheisyydessä (370km asti[5]). Senttimetrien tarkkuuteen päästään RKT-GPS (Real Time Kinematics GPS) -järjestelmällä, mutta tässä järjestelmässä korjausdataa lähettävän aseman tulee olla muutaman kilometrin läheisyydessä.

Satelliittipaikannusjärjestelmien ongelmana on huono kattavuus kaupunkiolosuhteissa. Etenkin suuremmissa kaupungeissa korkeat rakennukset häiritsevät satelliittien lähettämän signaalien vastaanottoa. Sisätiloissa GPS -paikannusta ei voida yleisesti käyttää. Kaupunkiolosuhteisiin on kuitenkin kehitetty tarkkuutta parantavia ratkaisuja. Esimerkiksi Japanissa kaupunkiolosuhteissa tehtävää paikannusta pyritään parantamaan QZSS (Quasi-Zenith Satellite System) -järjestelmällä [8]. Tässä järjestelmässä GPS -satelliittien lähettämään

[3] `http://www.glonass-center.ru/`
[4] `http://www.ngs.noaa.gov/FGCS/info/sans_SA/docs/statement.html`
[5] `http://en.wikipedia.org/wiki/Differential_GPS`

signaaliin yhdistetään QZSS -järjestelmän satelliittien lähettämä signaali. QZSS -satelliitit ovat havaittavissa keskitaivaan suunnalta ja muodostavat tiettyä aluetta seuraavan kiertoradan. GPS -järjestelmässä taas satelliitit ovat geostationäärisiä, eli ne sijaitsevat maasta katsoen aina samassa paikassa.

2.2.2 Verkkopaikannusjärjestelmät

Paikkatietoa voidaan tuottaa erilaisia langattomia tietoliikenneverkkoja hyväksikäyttämällä. Tukiasemapohjaisissa tietoliikenneverkoissa päätelaitteet kommunikoivat paikallaan olevien tukiasemien kanssa, jotka välittävät päätelaitteen lähetykset yleensä langallisesti muihin tietoliikenneverkkoihin. Tällaisia verkkoja ovat esimerkiksi GSM (Global System for Mobile Communications) -verkot. GSM -verkoissa sijaintitietona voidaan käyttää esimerkiksi päätelaitteen tiettynä hetkenä käyttämää tukiasemaa. Samaan tapaan voidaan kerätä myös WLAN -verkoista sijaintitietoa.

Tarkempaa paikkatietoa verkkopohjaisissa paikannusjärjestelmissä saadaan ottamalla huomioon esimerkiksi tukiasemien antennien kulmat tai signaalin etenemiseen käyttämä aika. Tällä tavoin päätelaitteen etäisyys ja suunta tukiaseman suhteen voidaan selvittää. Paikan määrityksessä voidaan käyttää myös havaittua signaalivoimakkuutta eri tukiasemista. Esimerkiksi Ekahau -yrityksen [9] EPE (Ekahau Positioning Engine) -tuote käyttää hyväksi signaalivoimakkuusinformaatiota WLAN -verkoissa tapahtuvaan paikannukseen. Tuote perustuu WLAN -päätelaitteen havaitsemiin signaalivoimakkuuksiin ja näiden korrelaatioon ennalta laadittuun signaalikarttaan. EPE:n avulla luodaan aluksi signaalivoimakkuuskartta paikannusalueesta. Signaalikartan pohjana käytetään paikannusalueena toimivan rakennuksen pohjapiirustusta. Malliin voidaan liittää useampia pohjapiirustuksia kuvaamaan rakennuksen eri kerroksia. Malliin määritetäään loogiset kulkureitit rakennuksessa “raiteiden” avulla. Kulkureittien määrityksen tarkoituksena on suurentaa paikannuksen tarkkuutta painottamalla paikannustuloksia raiteiden avulla.

Signaalikartan muodostaminen tapahtuu keräämällä signaalinäytepisteitä paikannusalueelta. Tämä tapahtuu kiertämällä haluttu alue esimerkiksi kannettavan tietokoneen avulla. Halutun näytepisteen sijainnissa osoitetaan nykyinen sijainti kartalta ja annetaan ohjelmiston kerätä signaalivoimakkuusnäyte (RSSI (Received Signal Strength Intensity)) sillä hetkellä kuulluista WLAN -tukiasemista. Näyte tallentuu tietokantaan, jota myöhemmin käytetään paikannusvaiheessa. Paikannusvaiheessa päätelaitteet mittaavat signaalivoimakkuutta kuulemistaan tukiasemista ja lähettävät näytteet palvelimelle, joka laskee näytteiden perusteella päätelaitteen sijainnin pohjakartalla. EPE soveltuu käytettäväksi sisätiloi-

hin ja sillä voidaan päästä jopa yhden metrin tarkkuuteen. EPE -palvelin tarjoaa sekä Java RMI (Remote Method Invocation) rajapinnan sekä TCP (Transmission Control Protocol) -protokollaa käyttävän rajapinnan paikannettujen päätelaitteiden paikkatiedon kyselyyn. Esimerkiksi Java -rajapinnan kautta paikkatietoa on mahdollista saada kahden sekunnin välein päivitettyinä näytteinä.

Muita esimerkkejä verkkopohjaisista paikannusjärjestelmistä on esimerkiksi Active Badge -järjestelmä [1], jossa käyttäjiä paikannettiin kannettavien infrapunalähettimien avulla. Infrapunalähetin sisällytettiin kannettaviin rintamerkkeihin. Rintamerkkien lähettämää signaalia vastaanottavia antureita kiinnitettiin huoneisiin ja yleisiin tiloihin. Antureita monitoroitiin työasemilla, jotka kyselivät antureilta havaintoja infrapunalähettimistä. Havainnot kerättiin talteen keskusasemalle, josta infrapunalähetimien sijaintia voitiin kysellä.

Edellä esitetyistä verkkopaikannusjärjestelmistä Active Badge on erityisesti paikannuskäyttöön suunniteltu. Ekahaun korrelaatiopaikannusta käyttävä järjestelmä taas käyttää hyväkseen olemassaolevia WLAN -verkkoja ja on täysin ohjelmistopohjainen ratkaisu. GSM -verkoissa on mahdollista pyrkiä parempaan tarkkuuteen lisäämällä erillisiä paikannuskäyttöön tarkoitettuja mittausyksiköitä. Esimerkiksi EOTD (Enhanced Observed Time Difference) -menetelmässä paikannus perustuu tukiasemien lähettämien signaalien aikaerojen havainnointiin[7]. Tätä menetelmää käytettäessä operaattorin verkkoon jouduttaisiin hankkimaan erillisiä LMU (Location Measurement Unit) mittausyksiköitä. Lisäksi verkkopaikannusjärjestelmissä paikkatiedon välittämiseen tarvitaan myös palvelu ja siten palvelinohjelmisto.

2.2.3 Lähipaikannusjärjestelmät

Lähipaikannuksella tarkoitetaan järjestelmiä, joissa paikkahavainnon hankkimisen kantama on lyhyempi kuin edellä esitetyissä tekniikoissa. Lähipaikantamiseen on esitetty käytettävän esimerkiksi radiotaajuustunnisteita (RFID (Radio Frequency Identification)). RFID -tunnisteissa on mikropiiri, johon on tallennettu tunnisteen sisältämä tieto, yleensä sarjanumero. Mikropiirin lisäksi tunnisteessa on antenni, jonka avulla mikropiirin sisältämä tieto voidaan lukea. RFID -tunnisteet voidaan jakaa aktiivisiin ja passiivisiin. Aktiivisissa tunnisteissa on usein muistin lisäksi älykkyyttä ja kyky lähettää tietoa. Passiiviset tunnisteet saavat mikropiirin toimintaan tarvittavan energian induktiosilmukkana toimivasta antennista. Tunnisteen lukijalaite tuottaa tällöin tunnisteen antenniin mikropiirin tarvittavan energian. RFID -tekniikkaa käytetään useimmiten tuotelähetyksien seurannassa ja tunnistamisessa, mutta tekniikkaa voitaisiin käyttää paikantamisessa. Esimerkiksi

Nokialla on olemassa matkapuhelinmalleja, joihin on saatavissa RFID -tunniste lisälaitteena. RFID -tunnisteiden käyttö olisi verrattavissa esimerkiksi Active Badge -tunnisteisiin tai Ekahau -yrityksen kannettaviin tunnisteisiin. Erona on käytetty radiotekniikka, RFID -tunnisteiden lukuetäisyys vaihtelee taajuusalueen mukaan (kosketusetäisyydestä eteenpäin). Lisäksi tunnisteiden taajuusalueet vaihtelevat, eikä ole olemassa yhtä standardia, laajasti käytettyä tunnistetekniikkaa.

Lähipaikannukseen voidaan luokitella myös Bluetooth -verkkoteknologiaa käyttävä paikannus. Bluetooth on yleistymässä lyhyen kantaman radiotekniikkana, jonka pääkäyttötarkoitus on toimia johtimien korvaajana. Bluetooth -standardiin on kuitenkin sisällytetty määritys paikannuskäytöstä LPP (Local Positioning Profile) profiilin avulla [10]. LPP -profiilissa määritetään protokolla sekä käytäntö paikkatiedon välittämiseen Bluetooth -laitteiden kesken. LPP -määrityksessä Bluetooth -laitteet voisivat hyödyntää toisia laitteita oman sijaintinsa määrittämiseen, mikäli laite itse ei pysty määrittämään sijaintiaan. Mikäli lähistöllä oleva laite pystyy vaikkapa GPS:n avulla määrittämään sijaintinsa, voi tämän laitteen sijaintia kyselevä laite päätellä yksinkertaisimmassa tapauksessa olevansa 10 metrin päässä tästä havainnosta. Tarkempaan etäisyyden päättelyyn jouduttaisiin soveltamaan monimutkaisempia algoritmeja, mahdollisesti samantyyppisiä kuten verkkopaikannuksessa (esimerkiksi antennikulmat, etenemisaika, signaalikartat ja korrelaatio). Pelkästään signaalivoimakkuuden käytön on todettu olevan riittämätöntä etäisyyden päättelyssä [11].

LPP:ssa laitteet ovat joko asiakkaita tai palvelimia ja ne voivat kysellä ja välittää paikkatietoa LPP:ssa määritellyin keinoin. Bluetooth -verkkoa on sovellettu paikannuskäyttöön mm. Raine Koposen diplomityössä [12], jossa toteutettiin Bluetooth -verkkoa käyttävä opastusjärjestelmä. Diplomityössä toteutettiin Bluetooth -tukiasemaverkon avulla opastesovellus, joka koostui kolmesta komponentista: Bluetooth -tukiasemissa olevista opaste -ohjelmista, keskusopas -palvelimesta sekä asiakkaana toimivasta asiakas-opas ohjelmistosta, jota käytettiin kämmentietokoneella. Asiakas-opas -ohjelman avulla käyttäjä pystyi kyselemään reittiä rakennuksen sisällä paikasta toiseen. Paikkatietona käytettiin kulloinkin yhteydessä olevaa tukiasemaa (solupaikkatieto). Työssä todettiin, että ongelmaksi muodostuu tukiasemien kattoalue. Ei voida olla varmoja käyttäjän oikeasta sijainnista, mikäli tukiasemien kuuluvuusalue on laaja.

2.2.4 Sovelluskohtaiset paikkatiedon lähteet

Paikkatiedon lähteenä voivat toimia myös järjestelmät, joita ei ole suunniteltu paikantamista silmälläpitäen. Esimerkiksi kalenterijärjestelmiä on käytetty paikkatiedon lähteenä. Erään toteutuksen esittelevät Urs Hengartnet ja Peter Steenkiste [13]. Toteutetussa järjestelmässä paikkatietoa kerättiin sekä käyttäjän päätelaitteen (kannettava tietokone WLAN -radiolla tai GPS -vastaanotin) paikantavan sovelluksen että kalenteriohjelmiston avulla. Kalenterissa olevien tapaamisten perusteella voitiin päätellä käyttäjän paikka ilman erillistä paikannusjärjestelmää. Julkaisussa esitetään myös tietoturvan kannalta tärkeä havainto: paikannusjärjestelmät eivät välttämättä ole yhden tahon hallinnassa. Paikkatiedon välityksessä joudutaankin siis luottamaan paikkatietoa välittäviin tahoihin. Myös paikkatiedon käytön tulisi olla käyttäjän hallinnassa. Käyttäjän tulee voida päättää mikä palvelu tai ketkä käyttäjät voivat saada hänen sijaintinsa tietoonsa.

Useasta eri järjestelmästä hankitun paikkatiedon hallittua yhteiskäyttöä kutsutaan "sensor fusion" -tekniikaksi [14]. Paikkatiedon hankkiminen ja mahdollinen yhdistely eri lähteistä johtaa paikannusjärjestelmän hajauttamisen harkintaan. Keskitetyn paikannusjärjestelmän (esimerkiksi Ekahau) ongelmana on paikkatiedon saannin varmuus. Mikäli keskitetty järjestelmä ei ole toiminnassa, ei paikkatietoa ole saatavilla. Hajautetussa järjestelmässä saatavuus taas voitaisiin jakaa eri lähteisiin. Paikkatiedon määritys vaihtelee järjestelmästä toiseen. Esimerkiksi GPS -paikannuksessa paikan määritys tapahtuu päätelaitteessa itsessään, kun verkkopaikannuksessa taas paikan määritys tehdään verkkoninfrastruktuurin elementissä. Ideaalitilanteessa saavutetaan parempi saatavuus paikkatiedolle käyttämällä useampaa paikanmäärityslähdettä. Tämä ei kuitenkaan aina ole mahdollista, esimerkiksi päätelaitteen rajoitusten takia. Esimerkiksi moniradioiset laitteet ovat harvinaisempia ja akun kesto rajoittaa radiolaitteiden käyttöä.

2.2.5 Paikkatiedon palvelurajapinnat

Jotta paikkatietoa voitaisiin käyttää erilaisissa sovelluksissa, tulee se hankkia ja välittää sovellusten käyttöön. Sovelluskehitystä edistävät tällöin rajapinnat, jotka määrittelevät protokollat ja menettelytavat, joilla paikkatietoa voidaan hankkia. Palvelurajapintojen avulla voidaan yleistää sovelluskehitystä ja tehdä sovelluskehityksestä määritellympää.

Paikkatiedon hankintaan on määritelty erilaisia rajapintoja monien eri tahojen toimesta. GPS -vastaanottimien tarjoamaa paikkatietoa voidaan lukea vastaanottimesta riippuen

esimerkiksi NMEA (National Marine Electronics Association) -standardin avulla[6]. NMEA -standardit määrittelevät sähköiset liitännät sekä tietotyypit, joilla erilaiset merenkulkulaitteet välittävät tietoa.

Verkkopaikannusjärjestelmistä paikkatiedon hakuun rajapintoja ovat määritelleet mm. OMA (Open Mobile Alliance)[7] ja Parlay[8]. OMA on matkaviestinnän alalla toimivien yritysten (operaattorit, laite- ja verkkovalmistaja sekä palveluntuottajat) yhteenliittymä, jonka tarkoituksena on luoda avoimia standardeja. Tällä tavoin pyritään nopeuttamaan ja yhteinäistämään matkaviestintäverkkojen palvelukehitystä. OMA:n toiminta on jaettu työryhmiin, jotka keskittyvät johonkin tiettyyn osa-alueeseen. Paikkatiedon hyödyntämiseen keskittyy LOC (Location Working Group)[9]. LOC jatkaa entisen LIF (Location Interoperability Forum) ja WAP (Wireless Application Protocol) Forumin työtä ja määrittelee standardeja paikkatiedon käyttöön matkaviestinverkoissa. LOC -työryhmä jatkaa mm. LIF:n MLP (Mobile Location Protocol) -protokollan määrittelyä. MLP on yleinen, verkkoteknniikasta riippumaton protokolla, jonka avulla voidaan kysellä mobiilien päätelaitteiden sijaintia. MLP:n määrityksessä langaton verkko sisältää palvelimen (Location Server) jolta paikkatietoiset sovellukset (Location Based Applications) tekevät kyselyjä MLP -protokollan avulla. MLP -protokolla on XML (Extensible Markup Language) -pohjainen. Käytettävää siirtokerrosta ei ole erikseen määritetty, mutta sidonta HTTP (HypertTex Transfer Protocol) -protokollalle on sisällytetty määrityksiin.

Parlay on OMA:n kaltainen usean yrityksen yhteenliittymä, joka myös pyrkii määrittelemään avoimia rajapintoja sovelluskehitykseen erilaisissa verkkoympäristöissä. Parlay 5.0 -specifikaatio on jaettu 15 osaan, joista osassa 6 (Mobility) määritellään rajapinta paikkatiedon hakuun. Mobility -rajapinnassa on määritelty protokolla ja rajapinnan luokkarakenne UML (Unified Modelling Language):n avulla. Määrityksen mukana toimitetaan myös WSDL (Web Services Description Language) ja IDL (Interface Description Language) -dokumentit rajapintakuvauksesta.

Paikkatiedon käyttöön sovelluksissa laatii määrityksiä myös OGC (Open Geospatial Consortium)[10]. Myös OGC on yritysten, valtion virastojen sekä yliopistojen yhteenliittymä, joka kehittää julkisesti saatavilla olevia rajapintamäärityksiä. OGC ei ole keskittynyt pelkästään määrittelyihin, joita voitaisiin käyttää paikkatiedon hakemiseen joltakin palvelimelta. OGC on määritellut monia ohjelmistokehitystä tukevia palveluja, esimerkiksi koor-

[6] http://www.kh-gps.de/nmea.faq
[7] `http://www.openmobilealliance.org`
[8] `http://www.parlay.org/`
[9] `http://www.openmobilealliance.org/tech/wg_committees/loc.html`
[10] `http://www.opengeospatial.org`

dinaattimuunnospalvelun [15].

2.2.6 GIS (Geographical Information System) -järjestelmät

Paikkatiedon (sijainti- ja ominaisuustieto) tehokkaaseen tallennukseen ja käsittelyyn käytetään usein GIS (Geographic Information System) järjestelmiä. Paikannussanastossa [3] GIS -järjestelmä on määritelty seuraavasti:

> "tietojärjestelmä, joka käsittelee paikkatietoa ja tukee erityisesti sijaintitiedon (1) käsittelyä ja hallintaa "
>
> "Paikkatietojärjestelmä tukee sijaintiin perustuvaa indeksointia, hakuja, analyysejä ja visualisointia."

Esimerkki GIS -järjestelmästä on GRASS (Geographic Resources Analysis Support System) -ohjelmisto[11]. GRASS -ohjelmistolla voidaan mm. visualisoida erilaista paikkatietoon liittyvää dataa. Yksi esimerkki tällaisesta tiedon käsittelystä on WLAN -verkon kuuluvuuden visualisointi. Visualisoinnin lisäksi paikkatiedon tallennusmenetelmät ovat tärkeässä asemassa tiedon määrän kasvaessa. Tallennukseen käytetään erilaisia tietokantajärjestelmiä jotka mahdollistavat myös tiedon haun. Haut voivat kattaa erilaisia operaatioita, joilla paikkahakua voidaan rajata. Paikkatietokantajärjestelmiä on toteutettu mm. relaatiotietokantajärjestelmiin. Esimerkiksi PostGIS [16] on eräs GIS -tietokantatoteutus, joka laajentaa PostgreSQL [17] olio-relaatiotietokantaa sisältämään tietotyyppimääreet paikka- ja sijaintitiedolle. Relaatiotietokantojen etuina on SQL (Structured Query Language) -kyselykieli, jolla voidaan tehdä hakuja tietokannan sisältöön.

2.3 Tilannetieto

Vaikka paikkatiedon on tietyltä osin hyvin määriteltyä, sama ei päde tilannetietoon. Tilannetiedolla tarkoitetaan yleisesti erilaisiin sovellusten käyttötilanteisiin liittyvää tietoa. Esimerkiksi Dey [18] on määritellyt tilannetiedon seuraavasti:

> "Konteksti on mitä tahansa informaatiota, jolla voidaan luonnehtia jonkin olion tilannetta. Olio on henkilö, paikka tai objekti, jonka katsotaan olevan merkityksellinen käyttäjän ja sovelluksen vuorovaikutukseen, mukaanlukien käyttäjä ja sovellus itse."

[11] `http://grass.ibiblio.org/index.php`

Käyttötilanteet voivat etenkin mobiilisovelluksissa vaihtua usein. Samaa sovellusta voidaan käyttää monessa eri tilanteessa ja ympäristössä. Sama toiminto voidaan tehdä myös monella eri sovelluksella eri ympäristöissä. Tiedettäessä sovelluksen käyttöympäristö, voidaan sovelluksen toimintaa muokata käyttöympäristön mukaiseksi. Vaihtoehtoisesti voidaan muokata eri ympäristöjä käyttäjän toimintoihin sopivaksi.

Ongelmana on kuitenkin tilannetiedon muoto. Erilaisia käyttötilanteita on monia, joten ennalta ei voida tietää mitä tietoa ja missä muodossa käyttötilanteesta on saatavilla. Etenkin mobiilisovelluksia voidaan käyttää monissa eri tilanteissa. Käyttötilanteeseen reagointi voidaan jättää joko käyttäjän tehtäväksi tai pyrkiä toteuttamaan se sovellukseen. Tilannetietoisia sovelluksia luotaessa on voitava määrittää, miten sovellus toimii eri tilanteissa. Tämä voi johtaa siihen, että sovellus räätälöidään ottaen huomioon vain halutut tilannetiedot, esimerkiksi paikkatieto. Monimutkaisemmissa tilannetiedon yhdistelmissä (esimerkiksi aika, paikka, säätila) tarvitsee sovellus jonkinlaisen ehdollisen tavan käsitellä saatavissa olevia tietoja. Esimerkiksi tiettyjen ehtojen täyttyessä tunnistetaan käyttötilanne ja suoritetaan toiminto sovelluksessa. Ongelmaksi voi tällöin tulla sääntöjen määrittely. Annetaanko käyttäjän määritellä toimintasäännöt vai jätetäänkö tämä sovelluskehittäjän tehtäväksi?

Tilannetieto voi koostua erilaisesta ympäröivästä tiedosta. Tällaista tietoa on esimerkiksi vallitseva säätila. Toisaalta tilannetieto voi olla käyttäjään henkilökohtaisesti liittyvää tietoa, esimerkiksi käyttäjän sijainti. Tilannetiedon käytössä tulisikin ottaa huomioon käyttäjän yksityisyyden suoja. Esimerkiksi paikkatiedon käsittelyä on lainsäädännössä rajoitettu. Erityisesti tilannetiedon käsittelystä ei ole säädetty vielä lakeja. Mutta tilannetiedon luonteesta johtuen, pitäisi noudattaa niitä säädöksiä mitkä käytettyä tietoa koskevat.

2.3.1 Tilannetiedon lähteet

Tilannetiedon lähteinä voi toimia yleisesti ottaen mikä tahansa sensori tai yleiskäyttöinen data. Sensorit ovat laitteita, jotka mittaavat ja aistivat jotakin ympäröivästä maailmasta. Esimerkiksi Dey:n [18] määritelmän perusteella tilannetiedon lähteiden määrittely onkin melko laajaa. Tämä johtaa siihen, että tilannetietoa käsittelevät järjestelmät ovatkin yleisiä työkaluja. Esimerkiksi MUPE -alustalle (esitelty kappaleessa 2.4) rakennettu tilannetiedon käsittelyrajapinta CEP (Context Exchange Protocol) määrittelee vain yleiset numeraaliset sekä merkkipohjaiset tietotyypit ja näiden koostumuksellisen käytön.

Tilannetiedon lähteenä voi toimia esimerkiksi vallitsevaa lämpötilaa mittaava sensori tai tätä tietoa välittävä sääpalvelu (esim. Ilmatieteen laitoksen sääpalvelu[12]. Tilannetiedon lähteenä voi toimia paikkatietoa tuottava järjestelmä, joka raportoi käyttäjän liikkeistä tietyllä alueella. Esimerkiksi GPS -vastaanottimella varustettu päätelaite voi tuottaa paikkatietoa käyttäjän liikkeistä ja tätä paikkatietoa voidaan käyttää hyväksi sovelluksen suorituksen yhteydessä. Tilannetietoisena sovelluksena ei kuitenkaan yleisesti käsitetä vain yhtä tiettyä tiedon lähdettä käyttävää sovellusta. Esimerkiksi kuljetun reitin seurantasovellusta ei yleisesti pidetä tilannetietoisena sovelluksena, vaikka käyttäjän kulkureitin seuraaminen, ja sitä kautta käyttäjän tilanteen seuraaminen, onkin tärkeässä asemassa.

Tilannetiedon lähteenä voi toimia joko yksittäinen sensori tai palvelu. Palvelut, kuten säätietopalvelut, keräävät ja jalostavat tilannetietoa. Tilannetiedon käyttöön voi liittyä myös rajoituksia, joita palvelut asettavat. Esimerkiksi paikkatiedon hakuun tietystä palvelusta voi liittyä käyttörajoituksia. Tällainen rajoitus voi olla vaikkapa käyttäjän suostumus hänen paikkatiedon käyttöön.

2.3.2 Rajapinnat ja sovelluskehitys tilannetiedolle

Erityisesti tilannetiedon käyttöön tarkoitettuja rajapintoja on vielä vähän. Tämä johtunee osaksi siitä, että tilannetiedon käyttö sovelluksissa ei ole vielä suuresti yleistynyt. Vielä ei olla varmoja onko tällainen yleinen tiedon hyödyntämistapa sovelluskehityksen yhteydessä hyödyllinen. Lisäksi ei olla varmoja ohjelmistokehityksessa tilannetiedon huomioonottamistavoista.

Tutkimuksen saralla tilannetietoa on kuitenkin hyödynnetty laajalti. Esimerkkejä löytyy Carnegie Mellon -yliopiston Aura -projektista[13]. Projektin konseptivideossa (saatavilla projektin etusivuilta) visioidaan ympäröivien tietokoneresurssien käyttöä aina tilanteeseen sopivimmalla tavalla. Tämä tehdään siten, että käyttäjän huomio säilyy itse käsiteltävässä asiassa, eikä pelkästään uuden tekniikan hallinnassa. Tällä tavoin hyödynnettiin ympäröiviä, erilaisia tietoteknisiä laitteita. Esimerkiksi käytävällä olevassa videotaulussa voidaan näyttää käyttäjän ohikulkiessa käyttäjää kiinnostavaa tietoa. Tilannetiedon käytöllä luodaan kokonainen järjestelmä, joka hallinnoi ympäröiviä tietokoneresursseja käyttäjän auttamiseksi.

Tilannetietoa voidaan hyödyntää myös yhden tietyn sovelluksen toiminnan kustomointiin. Tämä vaatii, että tilannetiedon olemassaolo otetaan huomioon sovellusta rakennettaessa.

[12] `http://www.fmi.fi`
[13] `http://www.cs.cmu.edu/~aura/`

Tilannetiedon huomioonottaminen taas vaatii, että sovellukseen on ennalta tehtävä toiminallisia määrityksiä tilannetietoa varten. Tämäntyyppinen sovelluksen toiminnallisuuden laajentaminen olisi monesti turhan paljon sovelluskehitystä rasittavaa, ellei olemassa ole työkaluja tilannetiedon käsittelyn helpottamiseksi. Erään tällaisen tilannetietoisten sovellusten kehitystyökalun tarjoaa MUPE -ympäristö (esitelty tarkemmin kappaleessa 2.4).

Tilannetietoisten sovellusten prototypointi olisi useasti vaikeaa ilman työkaluja tällaiseen tehtävään. Esimerkiksi paikkatiedon simulointiin jouduttaisiin rakentamaan työkaluja ennen kuin sovelluksen kehittäminen voidaan aloittaa. Työkaluja prototypointiin on kehitetty mm. Topiary -projektissa[14]. Topiaryn avulla voidaan projektin WWW -sivujen mukaan tehdä seuraavaa:

> Topiary on työkalu paikatietoisten sovellusten prototypointiin. Topiaryn avulla suunnittelijat voivat luoda kartan, joka mallintaa ihmisten paikkojen ja tavaroiden sijainnin; käyttää tätä karttaa demonstroimaan skenaarioita jotka kuvaavat paikan tilannetietoa; käyttää näitä skenaarioita tarinoiden luomisessa, jotka kuvaavat vuorovaikutussekvenssejä; ja ajaa näitä tarinoita mobiileissa laitteissa, kuten PDA -laitteissa, velho -sovelluksen avulla jolla päivitetään ihmisten ja tavaroiden sijainteja eri laitteella.

Tilannetiedon simulointi on tärkeää sovelluskehityksen testausvaiheessa. Simulointityökaluilla voidaan toimintaa testata ilman varsinaista kohdeympäristöä. Tällöin toiminnan testaaminen on nopeampaa. Tärkeää olisi kuitenkin myös se, että simulointityökalut olisivat yleiskäyttöisiä, eivätkä rajoittaisi käyttömahdollisuuksia vain simulointialustalle. Tällöin avoimet rajapinnat ovat hyvinkin tärkeitä.

Topiaryn lisäksi tilannetietoisten sovellusten kehittämistä on tutkittu myös Context Toolkit -projektissa[15]. Seuraava lainaus projektin WWW -sivuilta kertoo Context Toolkit -sovelluskehitysympäristön tarkoituksen:

> Context Toolkit koostuu context widgeteistä ja hajautetusta arkkitehtuurista, joka sisältää nämä widgetit. Context widgetit ovat ohjelmistokomponentteja jotka antavat sovelluksille pääsyn tilannetietoon samalla kun ne peittävät tilannetiedon havainnoinnin yksityiskohdat.

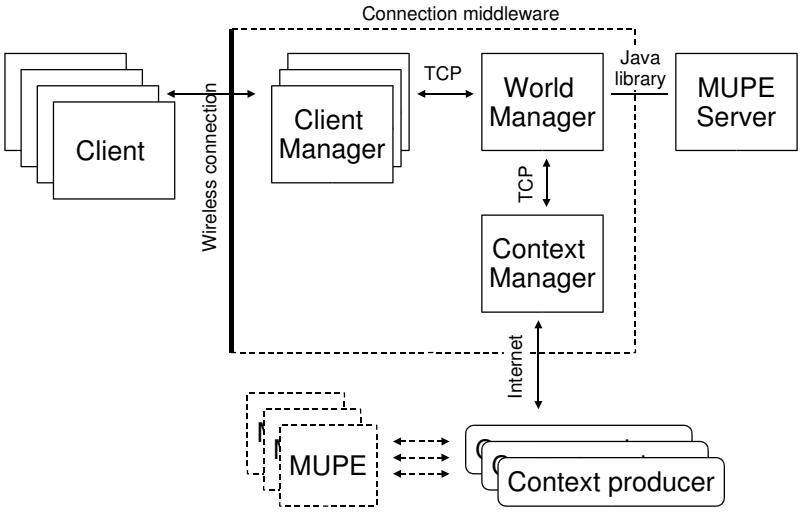
Context toolkit:n toimintaperiaate on samantyyppinen kuin graafisten käyttöliittymien kehityksessä kätettyjen yleisten komponenttien (engl. widget). Context toolkit tarjoaa komponentteja, joilla abstrahoidaan tilannetiedon keräämiseen, tallentamiseen ja pääsynhal-

[14] `http://guir.berkeley.edu/projects/topiary/`
[15] `http://www.cs.berkeley.edu/~dey/context.html`

lintaan liittyviä operaatioita, siten että huomio pysyy itse ohjelmiston kehityksessä. Graafisissa käyttöliittymissä käytetään usein yleisiä komponentteja esim. dialogien rakentamiseen, jotta näitä komponentteja ei jouduttaisi toteuttamaan erikseen jokaiseen graafiseen sovellukseen.

2.4 MUPE (Multi User Publishing Environment) -alusta

MUPE (Multi User Publishing Environment) -ympäristö on monen käyttäjän mobiilisovellusten kehittämiseen tarkoitettu alusta. Se koostuu asiakas- ja palvelinosasta, jotka kommunikoivat keskenään XML -rakenteisten viestien avulla. Kuvassa 2.1 on esitetty MUPE -alustan komponenttirakenne.



Kuva 2.1: MUPE (Multi User Publishing Environment) -alustan komponenttien yleiskuva.

Asiakasohjelma on rakennettu käyttäen J2ME (Java 2 Micro Edition) -ympäristön MIDP (Mobile Information Device Profile) -kirjastoja. MIDP -kirjastojen versioille 1 ja 2 on oma asiakasohjelmisto. Asiakasohjelma toiminta on selaintyyppistä; se tulkitsee XML -

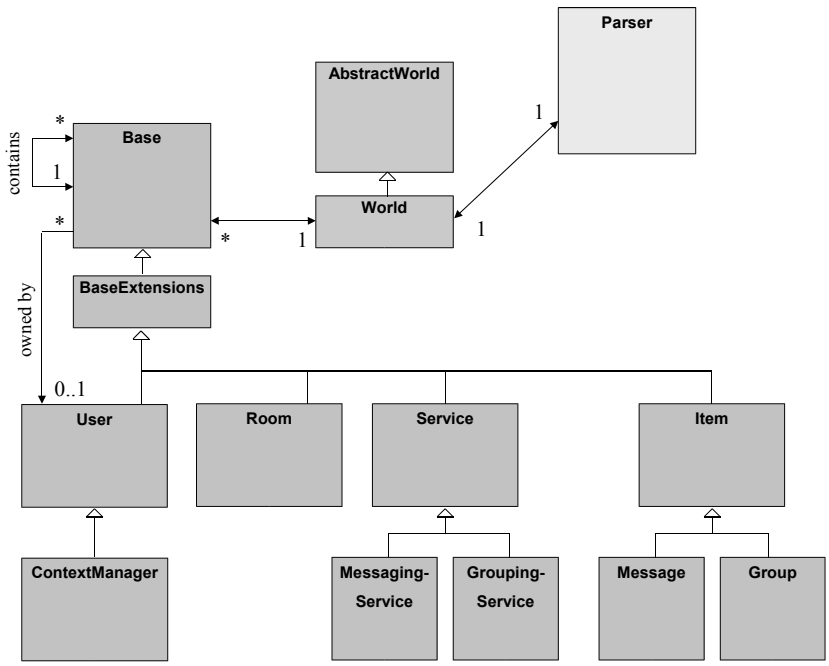
kuvauskielellä muodostettuja käyttöliittymäkuvauksia ja luo kuvauksen mukaisen käyttöliittymän toimintoineen. Palvelinosa koostuu sekä väliohjelmisto (middleware) -komponenteista että sovelluslogiikan sisältävästä komponentista. Kuvassa 2.1 esitetyt "middleware core" -komponentit kommunikoivat keskenään TCP -yhteyksien avulla. *Client Manager* -komponentti huolehtii asiakkaiden yhteyksien ylläpidosta. Tällä hetkellä *Client Manager* -komponentteja on sekä HTTP -protokollalle että MUPE:n omalle TCP -pohjaiselle protokollalle. *Context Manager* -komponentti huolehtii tilannetiedon vastaanottamisesta *Context Producer* -komponenteilta. Tilannetietoa välitetään TCP -yhteyden yli CEP -protokollan [19] mukaisissa XML -viesteissä. *World Manager* -komponentti huolehtii varsinaisen MUPE -sovelluksen logiikan suorituksesta.

MUPE -sovellus luodaan käyttämällä alustan tarjoamaa perusluokkarakennetta ja tämän rakenteen rajapintoja. Kuvassa 2.2 on esitetty MUPE -sovelluksen perusluokkarakenne. Luokkarakenteen luokkia *User*, *Room*, *Service* ja *Item* kutsutaan sisältöluokiksi (content classes) [20], sillä ne muodostavat MUPE -palvelun yleisen sisältörakenteen. Kyseinen luokkarakenne sisältyy kuvassa 2.1 esitettyyn *MUPE Server* -komponenttiin. Kuvan 2.1 *World Manager* ja *MUPE Server* komponenttien välinen rajapinta koostuu Java luokkarajapinnasta. Käytännössä *World Manager* -komponentti luo perusluokkarakenteen avulla objekteja MUPE -sovellukseen, esimerkiksi käyttäjät luodaan *User* -luokkaa käyttäen.

World Manager -komponentti käyttää *MUPE Server* -komponenttiin luotua luokkarakennetta *Parser* -luokan kautta. Parser -luokan toiminnallisuus koostuu XML -viestin tulkitsemisesta. Sekä *Client Manager*- että *Context Manager* -komponentit lähettävät *World Manager* -komponentille XML -viestejä, joita Parser -luokan avulla tulkitaan. XML -viesteillä voidaan suorittaa Java -metodikutsuja *MUPE Server* -komponentin sisältämästä sovelluksesta.

Luotaessa sovellusta määrittää *World* -luokka tietyille luoduille objekteille tunnistenumeron, jonka avulla luotuun objektiin voidaan viitata XML -viesteissä. Tunnuksen saavia objekteja ovat *Base* -luokasta perityt objektit. Tietyillä objekteilla on pysyvät tunnistenumerot. Näitä ovat *World* (numero 0), oletushuone (default room, numero 1) ja *Context Manager* (numero 2). Näillä kolmella objektilla on erityisrooli MUPE:n toiminnallisuudessa, esimerkiksi *Context Manager* -objektia vastaa *Context Manager* -komponentti. Kyseisen komponentin lähettämissä XML -viesteissä olevia Java -metodikutsuja kutsutaan vastaavan nimisestä objektista *MUPE Server* -komponentin sisällä [16]. XML -kutsuissa kerrotaan

[16] Alun perin MUPE -sovelluksen sisältämä *Context Manager* -objekti oli nimeltään *Superuser*. Kyseisellä objektilla oli valtuudet tehdä metodikutsuja kaikista MUPE -sovelluksen luokista. Myöhemmin osoittautui että *Superuser* -objektin käyttökohde oli pelkästään tilannetiedon käsittely, joten kyseinen objekti nimettiin *Context Manager* -objektiksi.



Kuva 2.2: MUPE (Multi User Publishing Environment) -alustan sovelluksen perusluokkarakenne

objekti, josta kyseinen kutsu tehdään. Esimerkki tällaisesta XML -kutsusta löytyy lähteestä [20]:

```
146645::clientAnswerCreator {“Bob”}
```

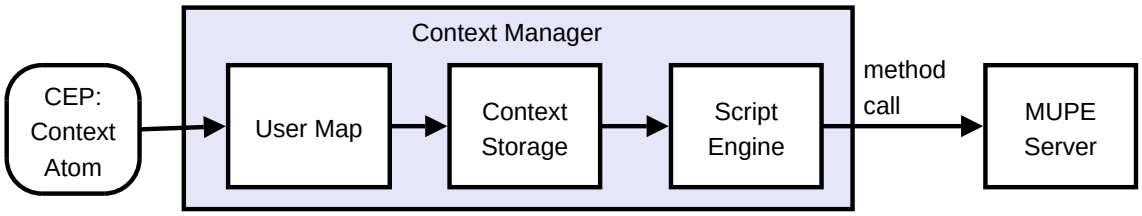
Kyseisessä esimerkissä tehdään *clientAnswerCreator* -metodikutsu objektiin, jonka tunnusnumero on *146645*. Kutsussa viedään parametrina merkkijono “Bob”. MUPE -sovellus vastaa XML -kutsuihin yleensä uudella XML -dokumentilla, joka lähetetään takaisin kutsun tehneelle asiakkaalle. Poikkeuksena on *Context Manager* -komponentti, jolle ei lähetetä tehtyjen kutsujen vastauksia.

Context Manager -komponentin avulla MUPE -sovelluksiin voidaan tuoda tilannetietoa ympäröivästä maailmasta. Tämä tapahtuu muokkaamalla haluttu tilannetieto CEP -protokollan mukaisiin XML -dokumentteihin. Tätä varten MUPE -alustalle joudutaan tekemään jokaista tilannetiedon lähdettä vastaava tilannetiedon tuottaja, *Context Producer*.

Tilannetiedon tuottajakomponentit toimivat TCP -palvelimina; ne vastaanottavat TCP -

yhteyksiä MUPE -sovelluksilta ja välittävät tuottamansa tilannetiedon CEP -protokollan mukaisina XML -viesteinä muodostetun TCP -yhteyden yli MUPE -sovelluksille. Tilannetiedon lähteestä riippuen tilannetiedon tuottajan toteutus vaihtelee. Periaatteessa tilannetiedon tuottaja toimii kuitenkin eräänlaisena "protokollamuuntimena", joka muuntaa tilannetiedon lähdejärjestelmän tietotyypit vastaamaan CEP -protokollan [19] määritystä. Tällöin MUPE -sovellukset voivat tulkita yleistä CEP -protokollan mukaista tietoa. MUPE -sovelluksen käynnistyessä määritetään *Context Producer* -komponentit, joihin otetaan yhteys IP -osoitteen ja portin perusteella. Tämän jälkeen *Context Producer* -komponentin tehtäväksi jää välittää tilannetiedon muutokset CEP -protokollan mukaisina XML -dokumentteina kytkeytyneinä oleville *Context Manager* -komponenteille. CEP -protokollan määritys [19] tukee monia operaatioita, mutta kaikkia näistä ei ole toteutettu tällä hetkellä MUPE -alustan *Context Manager* -komponentissa.

Context Manager -komponentissa voidaan vastaanotetun tilannetiedon perusteella suorittaa toimintoja MUPE -sovelluksen toimintalogiikassa. Vastaanotetun tilannetiedon prosessointi kulkee kolmessa vaiheessa, jotka on esitetty kuvassa 2.3. Tilannetiedon prosessoinnin eriyttäminen omaan komponenttiinsa mahdollistaa sen, että jokaiseen MUPE -sovellukseen ei tarvitse erikseen tehdä komponenttia tilannetiedon käsittelyyn.



Kuva 2.3: *Context Manager* -komponentin sisäisen toiminnan kaaviokuva. [21]

CEP -protokollan mukaisessa tilannetiedossa (*Context Atom*) voidaan määrittää olio, johon tilannetieto liittyy ([19] s. 10-11). Kuvassa 2.3 olevassa *User Map* -vaiheessa muutetaan oliotunniste sovelluksen käyttämäksi tunnisteeksi. *Context Manager* -komponentissa voidaan luoda tilannetiedon tuottajakohtaisesti muunnosmäärityksiä, jossa tietyn tilannetiedon tuottajan lähettämän tilannetietoa vastaavan olion tunniste muunnetaan vastaamaan sovelluksessa olevaa objektia. Esimerkki tällaisesta muunnoksesta on vaikkapa paikkatietoa käsiteltäessä laitteen tunnisteen muuttaminen laitetta käyttävän käyttäjän tunnisteeksi. Nämä muunnosmääritykset voivat sijaita tiedostossa, joka luetaan muistiin *Context Manager* -komponentin käynnistyessä. Vaihtoehtoisesti *World Manager* -komponentti voi lähettää *Context Manager* -komponentille XML -viestin, jossa määritellään uusi tai poistetaan vanha muunnos. Listauksessa 2.1 on esitetty esimerkki olion tunnisteen muunnos-

määrittelystä.

```
<addUserMap>
  <userMap>
    <contextProducerName>WLAN-paikannusjarjestelma
    </contextProducerName>
    <contextProducerId>00:02:2D:07:4A:14
    </contextProducerId>
    <serverId>Brian
    </serverId>
  </userMap>
</addUserMap>
```

Listaus 2.1: Oliotunnisteiden muunnosmääritys

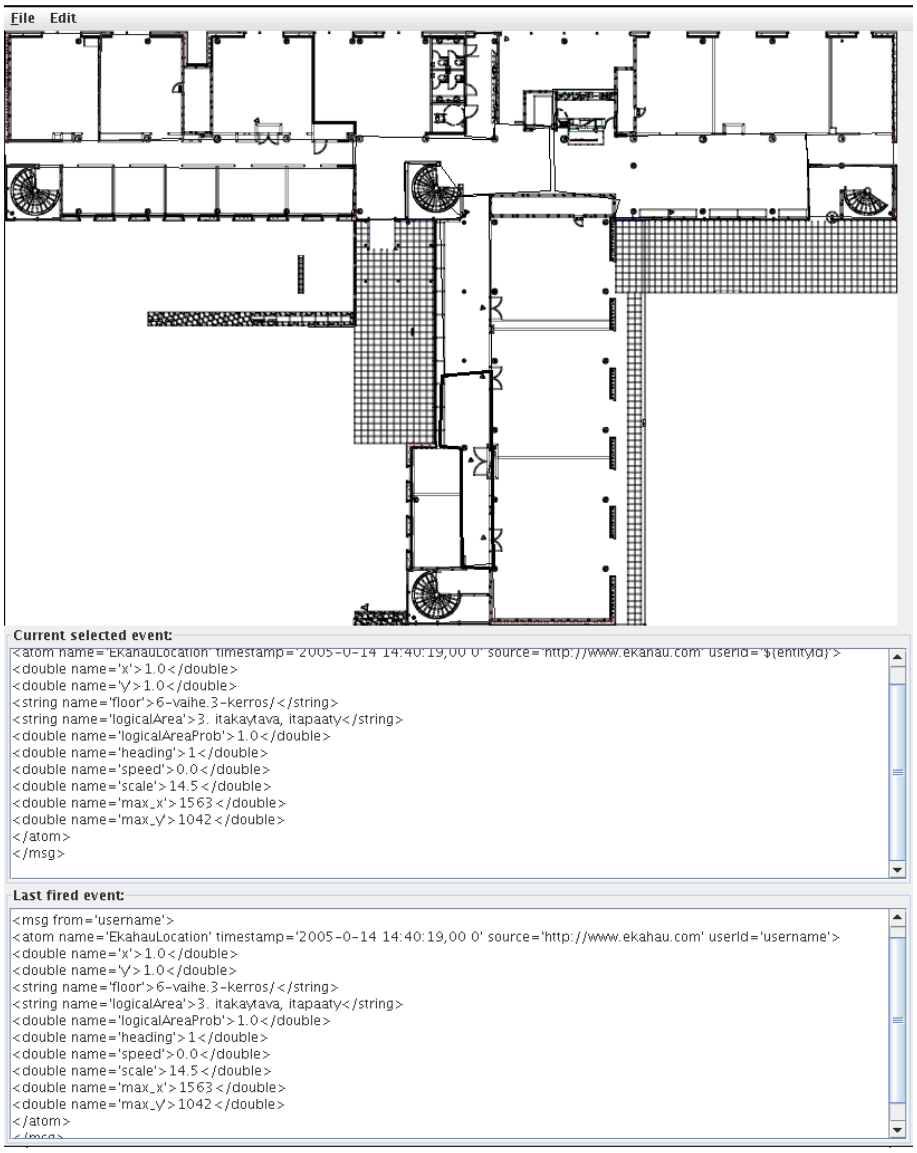
Oliotunnistemääritysten käsittelyn jälkeen tilannetieto tallennetaan *Context Storage* -komponenttiin. Tallennuksen yhteydessä ilmoitetaan *Script Engine* -komponentille uuden tilannetiedon saapumisesta. *Script Engine* -komponentti sisältää XML -muodossa määritettyjä skriptejä, joilla määritetään "if-then-else" -tyyppisesti ehtoja. Näissä ehdoissa voidaan viitata tallenettuihin tilannetietoihin (CEP -protokollan "Atomi"). Ehtojen täyttyessä voidaan laukaista toimintoja MUPE -palvelimesta. MUPE -alusta sisältää työkalut skriptien luomiseen. Listauksessa 2.2 on esimerkki skriptistä, jossa määritetään MUPE palvelimesta kutsuttavaksi metodikutsu *clientSetLocation* silloin kun CEP -atomissa nimeltä *LISLocation* tapahtuu muutos millä tahansa oliotunnisteelle.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<script id='lis'
        xmlns='http://www.nokia.com/ns/cep/script/1.0/'
        xmlns:cep='http://www.nokia.com/ns/cep/1.0/'>
  <if>
    <atomChanged>
    <atomRef userId='*' name='LISLocation'>
    </atomRef>
    </atomChanged>
      <actions>
        <mupeCall>
<![CDATA[2::clientSetLocation {${userId}} {${LISLocation::location}}]]>
        </mupeCall>
     </actions>
  </if>
```

```
</script>
```

Listaus 2.2: Esimerkki tilannetiedon perusteella toimintoja laukaisevasta skriptistä.

Tilannetietoa voidaan lähettää MUPE -sovelluksille *Context Producer* -komponettien lisäksi simulaattoreilla. Simulaattorit kommunikoivat *World Manager* -komponentin kanssa. Simulaattorien avulla voidaan muodostaa nopeasti CEP -protokollan mukaisia XML -dokumentteja, joiden sisältämiä CEP -protokollan mukaisia tietorakenteiden arvoja voidaan simulaattorin avulla vaihdella. MUPE -alustaan sisältyy graafinen työkalu simulaattorien tekoon. Simulaattorit on jaoteltu neljään luokaan: tapahtumapohjainen, aluepohjainen, 1D -jatkuva ja 2D -jatkuva. Tapahtumapohjaisella simulaattorilla voidaan rakentaa yksittäisiä tapahtumia simuloivia CEP -atomeja. Esimerkiksi kuun vaiheita voitaisiin simuloida tällä tavoin. Aluepohjaisella simulaattorilla voidaan laukaista tapahtumia määrittämällä tapahtumaa vastaava alue annetulta pohjakartalta. Tämäntyyppinen simulaattori sopii esimerkiksi huonekohtaista paikkatietoa tuottavan järjestelmän simulointiin. Yksiulotteista, jatkuvamuotoista tietoa tuottavalla simulaattorilla voidaan simuloida tietyn muuttujan vaihteluita määritetyltä arvoväliltä. Tällainen arvoväli on esimerkiksi lämpötila. Kaksiulotteista, jatkuvamuotoista dataa tuottavalla simulaattorilla taas voidaan simuloida esimerkiksi koordinaattimuutoksia annetulta pohjakartalta. Kuvassa 2.4 on esimerkki aluepohjaisen simulaattorin käyttöliittymästä. Simulaattori on jaettu kolmeen osaan. Ylimmäisessä osassa on simuloitavan alueen pohjakartta. Pohjakartalle on määritetty alueet, jotka vastaavat simuloitavia CEP -atomin tietoja. Toiseksi alimmaisessa osassa näytetään CEP -atomi, joka vastaa tällä hetkellä valittua aluetta (sama alue näytetään myös tummennetuilla reunoilla simulaattorin yläosassa). Alimmaisessa osassa näytetään viimeksi MUPE -sovellukselle lähetetty CEP -atomi. Simulaattoria voidaan käyttää vain samalla koneella, jossa itse MUPE -palvelin ajetaan, sillä *World Manager* -komponentti hyväksyy vain lokaaleja yhteyksiä.



Kuva 2.4: Esimerkki aluepohjaisesta simulaattorista.

Luku 3

Toteutetut palvelut

Tässä luvussa kuvataan toteutettuja järjestelmiä sekä palveluita, joissa kerätään ja hyödynnetään paikka- sekä tilannetietoa. Järjestelmien toteutuksessa on hyödynnetty Lappeenrannan Teknillisen Yliopiston (LTY) tietoliikenteen laitoksen WLPR.NET -projektissa rakennettua WLAN -verkkoa (myöhemmin WLPR.NET).

WLPR.NET kattaa LTY:n kampus -alueen, osan Lappeenrannan Skinnarilan ja Sammonlahden kaupunginosista sekä myös muutaman alueen kaupungin keskustasta. WLPR.NET -verkko on rakennettu *Lappeenranta* -mallin mukaisesti [22], joka tarkoittaa että WLAN -verkossa käyttäjät voivat vapaasti liittyä verkkoon. Verkko itsessään tarjoaa vain alueelliset yhteydet. WLPR.NET -verkko on operaattorineutraali, toisin sanoen, mikään yksittäinen verkko-operaattori ei kokonaan omista tai hallinnoi verkkoa. Internet -operaattoreita varten WLPR.NET -verkko tarjoaa yhdysliikennepistemallin, jonka kautta Internet -operaattori voi tarjota WLPR.NET -verkkoon Internet -yhteyttä. Käyttäjät voivat vapaasti valita operaattorin, jonka kautta heidän Internet -liikenteensä reititetään. Lisäksi WLPR.NET -verkko tarjoaa mahdollisuuden paikallisiin palveluihin, joita voidaan alueverkkoon rakentaa ilman että pääsy näihin palveluihin kulkisi Internet -operaattorin kautta. Voidaankin sanoa, että tällaisen verkkomallin rakentamisen perustana on ollut avoimen tietoyhteiskunnan kehitys, jossa tietoverkkoon pääsy on avointa ja kaikille mahdollista.

LIS -palvelun toimintaidea mukailee *Lappeenranta* -mallissa [22] esitettyä ideaa palvelurajapinnasta. Jotta WLPR.NET -verkkoon liittyvät palveluntarjoajat olisivat tietoisia verkon tarjoamista palveluista, tarjoaa verkko rajapintoja palveluntarjoajien käyttöön. Palvelurajapinta toimii myös paikkana, josta käyttäjät voivat nähdä verkossa toimivat palvelut. Verkon itsensä tarjoama palvelu on tässä tapauksessa paikannuspalvelu, joka pystyy kerto-

maan käyttäjien sijainnin verkon alueella. Paikkatietopalvelu on verkon itsensä tarjoama, mutta palvelurajapinnan tarkoitus on toimia myös pisteenä, jossa muutkin palveluntarjoajat, kuin verkko itse, voivat mainostaa palveluitaan.

WLPR.NET -verkkoon on kehitetty solupaikannuspalvelu, jolla kerättiin verkossa liikkuvien päätelaitteiden sijaintia tukiasemakohtaisesti. Myös muita paikannusmekanismeja oli tutkittu [23]. Kuitenkin ongelmana oli edelleen se, että tutkitut palvelut olivat jäänee pääosin verkon ylläpidon käyttöön (esim. Mikko Hietalan paikkatietoinen pikaviestinpalvelu [24]). Palvelut sijaitsivat verkon palvelimilla ja pääsy paikkatietoon vaati pääsyä suoraan itse tietokantaan, jonne solupaikkatieto oli kerätty. Tämän mallin ongelma on, että uusille kehitettäville palveluille jouduttiin sallimaan pääsy suoraan itse tietokantaohjelmiston pääsynhallintamekanismein (jotka sinänsä voivat olla hyvinkin päteviä), mutta palvelunkehitys ei näin ollen ollut vapaata. Ongelma on myös se, että suoralla tietokantayhteydellä paikkatietoinen palvelu on hyvin riippuvainen solupaikkatiedon tietomallista. Tämä tietomalli taas ei välttämättä ole sopiva itse palvelulle. Lisäksi, jos paikkatiedon keräykseen ja tietomalliin tehdään muutoksia, joudutaan samalla myös palvelua muuttamaan, vaikka tämä ei välttämättä olisi tarpeellista. Palvelut joutuvat sitoutumaan myös vain solupaikkatiedon käsittelyyn, vaikka paikkatiedon esitys karttamallilla voisi olla sopivampaa.

Ongelmaksi voidaan kokea tämän tyyppisessä integroidussa mallissa se, että paikkatiedon käsittelyssä jäivät monet parametrit huomioimatta. Esimerkiksi paikkatiedon tarkkuus, varmuus ja paikkatiedon käytön salliminen olivat kaikki asioita, jotka palvelujen tuli itse ottaa huomioon. Palvelunkehitystä helpottaisi, mikä nämä parametrit olisivat tarjottuina itse paikkatietopalvelusta käsin. LIS -palvelun kehittäminen nähtiin mahdollisuutena tuoda ratkaisu osaan näistä ongelmista. LIS ei kuitenkaan tarjoa ratkaisua kuin osaan näistä ongelmista, mutta muut osuudet on jätetty jatkokehityksen varalle.

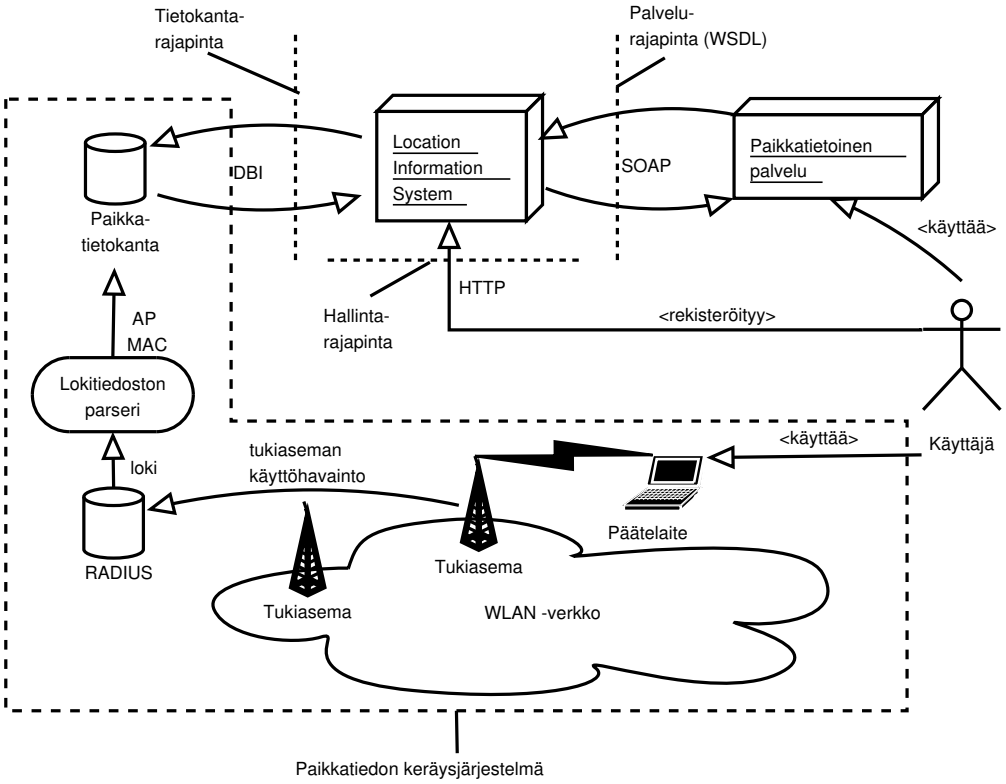
3.1 Location Information System - LIS

Location Information System (LIS) on palvelu, joka tarjoaa rajapinnan paikkatietoisille palveluille. Palvelun tarkoitus on tarjota rajapinta, joka mahdollistaa paikkatiedon käytön WLAN -verkossa. Palvelu mahdollistaa myös paikkatiedon käytön hallinnan sekä toimii yhteyspisteenä, jonka kautta käyttäjät voivat nähdä käytettävissä olevat paikkatietoiset palvelut. LIS -palvelun toimintaa on kuvattu julkaisussa [25] (Location Information System – a Description of a Positioning Middleware System).

LIS -palvelun arkkitehtuurin yleiskuva on esitetty kuvassa 3.1. LIS koostuu paikkatiedon

keruun muodostavasta järjestelmästä, palveluille tarjotusta pyyntörajapinnasta sekä palvelujen ja käyttäjien hallintarajapinnasta. WLAN -päätelaitteiden paikkahavainnot kerätään tietokantaan ja LIS tarjoaa pääsyn tähän tietoon SOAP (Simple Object Access Protocol) -rajapinnan kautta. SOAP -rajapinnan rakenne on kuvattu WSDL -rajapintakuvauskielen avulla. WSDL -dokumentti, jossa rajapinta on kuvattu, on saatavissa WLPR.NET -verkon WWW -sivujen kautta. LIS -palvelu on toteutettu käyttäen Perl -ohjelmointikieltä sekä SOAP::Lite -kirjastoa [26] SOAP -protokollan käyttämiseen.

SOAP -protokollan valinta käytettäväksi rajapinnaksi paikkatietokyselyille johtui suuresta sovelluskehitystyökalujen määrästä. SOAP -protokollan toteuttavia kirjastoja oli saatavilla useille eri ohjelmointikielille (mm. Perl, Java, C, C++). SOAP on osa Web Services -arkkitehtuuria[27], joten kyseisen protokollan valinta nähtiin hyvänä lähtökohtana verkkopalveluiden kehittämiseen.

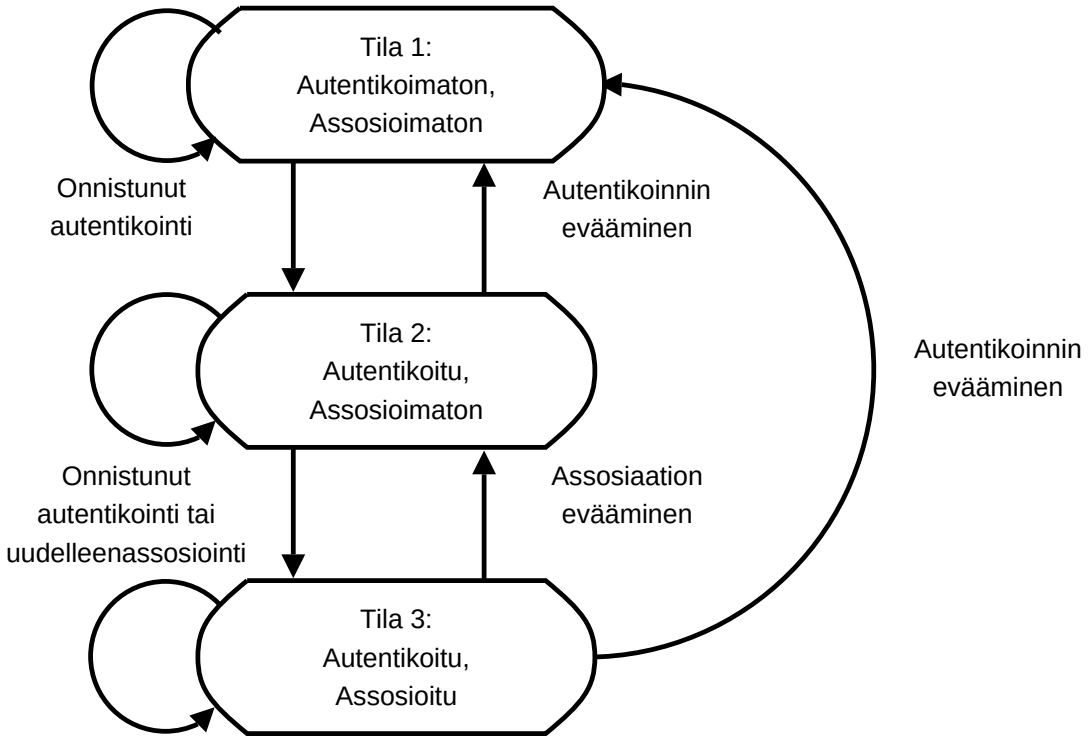


Kuva 3.1: LIS -yleiskuva.

3.1.1 Paikkatiedon keruu

LIS -järjestelmä käyttää WLAN -verkosta saatavaa solupohjaista paikkatietoa. WLAN -verkkona on käytetty WLPR.NET -projektin rakentamaa WLPR.NET -verkkoa. Paikkatietohavaintojen keruun pohjana toimii käyttäjien WLAN -tukiasemien käytön seuraaminen. Tarkempaa paikkatiedon keruujärjestelmää ei toteutettu, sillä solupaikannuksen hyödyntäminen nähtiin riittävänä. Toisaalta tarkemman paikannusjärjestelmän käyttäminen olisi vaatinut huomattavasti enemmän resursseja tai valmiin tuotteen (kuten Ekahau EPE) käyttämistä.

Kun käyttäjä muodostaa yhteyden WLAN -päätelaitteellaan (esimerkiksi kannettava tietokone, jossa on 802.11b -standardien mukainen WLAN -radio) WLAN -tukiasemaan, käy päätelaite läpi tilanvaihdokset, jotka on määritelty IEEE (Institute of Electrical and Electronics Engineers):n 802.11 -standardissa [28] (kpl 5.5 Relationships between services). Tilanvaihdokset on esitetty kuvassa 3.2.



Kuva 3.2: WLAN -päätelaitteen tilakaavio.

WLAN -päätelaitteet pitävät yllä kahta tilamuuttujaa, *assosiaatio* ja *autentikaatio*. Assosiaatiolla tarkoitetaan tilaa jossa on muodostettu yhteys WLAN -päätelaitteen ja tukiase-

man välillä ja päätelaite voi välittaa dataa WLAN -verkossa. Autentikaatiolla tarkoitetaan tilaa, joka ilmaisee onko päätelaitteella lupa liittyä WLAN -verkkoon ja välittää dataa verkossa.

Kun päätelaite liittyy verkkoon, autentikoituu laite ensin tukiasemalle. WLPR.NET -verkossa on käytössä 802.11 -standardissa määritetty *Open System Authentication* -autentikointitapa. *Open System Authentication* -autentikointitavassa ei ole käytössä varsinaista autentikointialgoritmia, vaan tukiasema joko sallii tai evää päätelaitteen autentikointipyynnön. Autentikointipyynnön sallimisen perustana käytetään WLPR.NET -verkossa RADIUS (Remote Authentication Dial In User Service) -palvelua. Verkossa olevat tukiasemat, joista suurin osa on Orinoco AP-1000 mallia, on konfiguroitu käyttämään autentikointiin RADIUS -palvelua. Kun päätelaite yrittää liittyä WLPR.NET -verkkoon, siirtyy se kuvan 3.2 mukaan tilasta 1 tilaan 2. Tällöin tukiasema lähettää autentikointipyynnön RADIUS -palvelimelle. Jos RADIUS -palvelin vastaa myöntävästi pyyntöön, antaa tukiasema päätelaitteelle ilmoituksen autentikoinnin onnistumisesta. Tämän jälkeen päätelaite voi alkaa käyttämään verkkoa. Tällöin tapahtuu assosiaatio tukiaseman kanssa. Myös päätelaitteen liikkuessa eri tukiasemien välillä, autentikoituu päätelaite uudelle tukiasemalle[1].

RADIUS -palvelu pitää lokia tehdyistä autentikointipyynnöistä. Näin ollen lokitiedostoon muodostuu päätelaitteiden sijaintihistoria. Nämä lokimerkinnät sisältävät autentikointipyynnön tehneen tukiaseman IP (Internet Protocol) -osoitteen sekä autentikoidun päätelaitteen MAC (Media Access Control) -osoitteen. Päätelaitteen liikkuessa verkon alueella tukiasemasta toiseen, voidaan laitteen liikkuminen päätellä lokin sisällöstä.

RADIUS -palvelimen lokia seurataan WLPR.NET -verkossa parseriohjelmalla, joka päivittää paikkatietohavaintoja kuvassa 3.1 esitettyyn tietokantaan. Tietokanta sisältää WLAN -päätelaitteiden viimeisimmät sijaintihavainnot. Tämä paikkatiedon keruumekanismi luottaa RADIUS -palvelun toimintaan edelläkuvatulla tavalla. Kuitenkin projektin aikana käyttöön otettiin tukiasemia, joilla ei ollut mahdollista hallita autentikointiprosessia RADIUS -palvelun avulla. Näitä tukiasemia varten (mm. Nokia A032 tukiasemat) on WLPR.NET -projektissa toteutettu kyselymekanismi, jolla väliajoin aktiivisesti kysellään tukiasemissa assosioituneena olevia päätelaitteita. Kyselymekanismi on toteutettu WLPR.NET -projektissa Perl -skriptillä, jossa on tukiaseman tyypistä riippuva toteutus assosioituneiden päätelaitteiden hakuun (esim. HTML (Hypertext Markup Language) -sivun tulkinta Nokia A032 tukiasemasta). Kyselymenetelmän ongelma on se, että tukiasemien siltaustauluissa

[1] 802.11 standardi määrittelee myös myös "esiautentikoinnin" (Preauthentication), jossa päätelaite autentikoituu havaitsemilleen tukiasemille etukäteen ennen assosiointia. Tämä menettely nopeuttaa tukiasemasta toiseen liikkumista (roaming), mutta voi myös haitata autentikointiin perustuvaa paikkatietokeräystä.

saattaa olla tukiasemaa käyttävien WLAN -päätelaitteiden MAC -osoitteita vielä senkin jälkeen, kun päätelaite on siirtynyt käyttämään toista tukiasemaa. Siirryttäessä käyttämään tukiasemaa, joka käyttää RADIUS -autentikointia, päivittyy uusi solusijaintitieto tietokantaan. Mutta siirryttäessä takaisin sellaisen tukiaseman alueelle, jossa RADIUS -autentikointi ei ole käytössä, saatetaan tulkita, että päätelaite olisi vielä vanhassa sijainnissaan ja päivitystä ei tehdä.

3.1.2 WWW -hallintarajapinta

WWW -hallintarajapinta toteutettiin PHP (PHP Hypertext Preprocessor) ohjelmointikielen avulla ja Apache 1.3 WWW -palvelinta käyttäen. Nämä työkalut valittiin sekä hyvän dokumentaation, että aikaisemman kokemuksen vuoksi. Käyttäjät hallitsevat paikkatietonsa käyttöä WWW (World Wide Web) -pohjaisen hallintarajapinnan kautta. Ennen kuin hallinta on mahdollista, rekisteröityvät käyttäjät LIS -palveluun. Rekisteröinnin avulla sidotaan käyttäjän identiteetti päätelaitteen WLAN -radioon. Rekisteröinnin toiminta on seuraava:

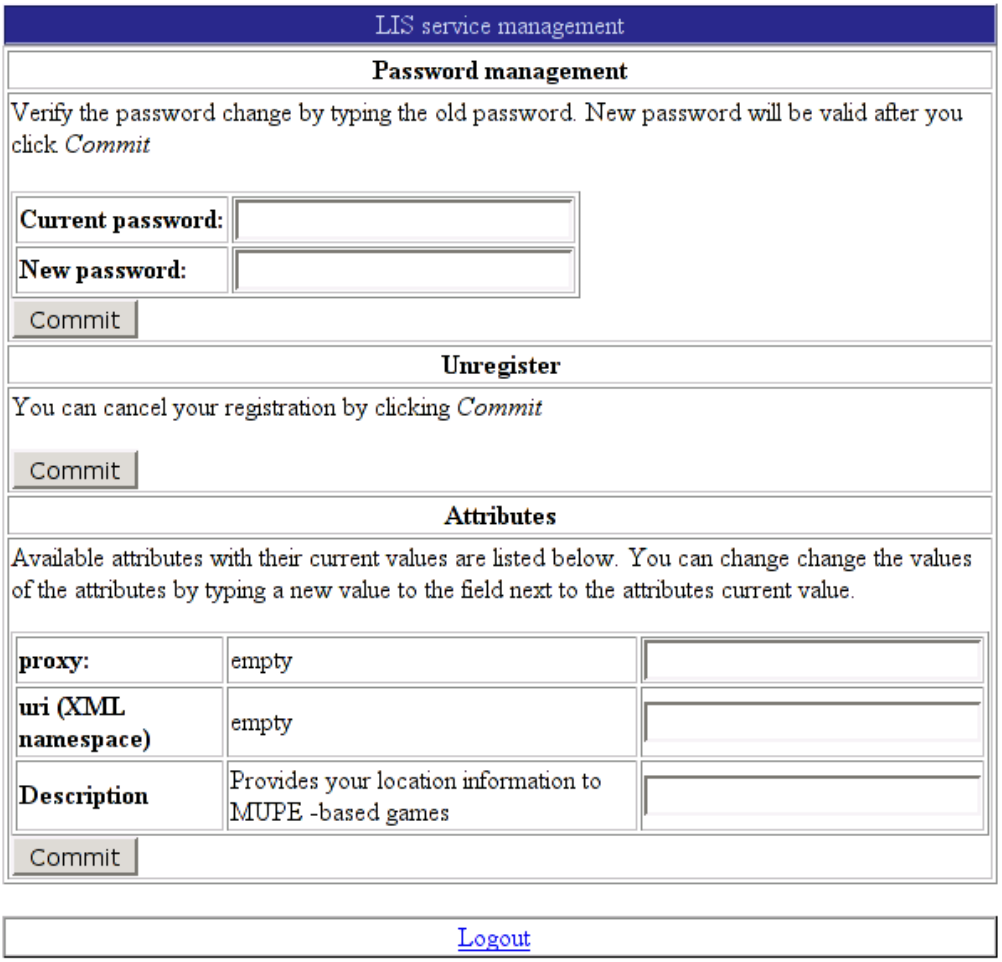
1. Käyttäjä ottaa WWW -selaimella yhteyden LIS -palvelun WWW -palvelimelle ja aloittaa rekisteröintiprosessin.
2. Rekisteröintiprosessissa otetaan talteen käyttäjän IP -osoite.
3. WWW -palvelimen ARP (Address Resolution Protocol) -taulusta etsitään IP -osoitetta vastaava MAC -osoite.
4. MAC -osoite tallennetaan käyttäjän profiiliin.

Rekisteröinnin jälkeen käyttäjä voi valita palvelut, joille hän myöntää luvan oman paikkatiedon käyttöön. Käyttäjä voi myös myöhemmin vaihtaa hänellä käytössään olevaa päätelaitetta ja ilmoittaa vaihdon LIS -järjestelmään. Käyttäjällä voi olla vain yksi laite kerrallaan käytössä.

Palveluntarjoaja, joka haluaa hyödyntää WLAN -verkon käyttäjien paikkatietoa, rekisteröi palvelunsa LIS -järjestelmään. Palvelut tunnistetaan tunnuksesta, jonka palveluntarjoaja voi rekisteröintivaiheessa määrittää. Palvelun rekisteröinnissä palveluntarjoaja jättää palvelustaan kuvauksen LIS -järjestelmään.

Kuvassa 3.3 on esitetty palveluntarjoajan käyttöliittymä. Palveluntarjoaja saa palvelutunnuksen rekisteröitymisen yhteydessä salasanan, joka voidaan vaihtaa käyttöliittymän toiminnolla. Samoin palvelu voidaan poistaa käyttöliittymästä käsin. Mikäli palvelu käyttää

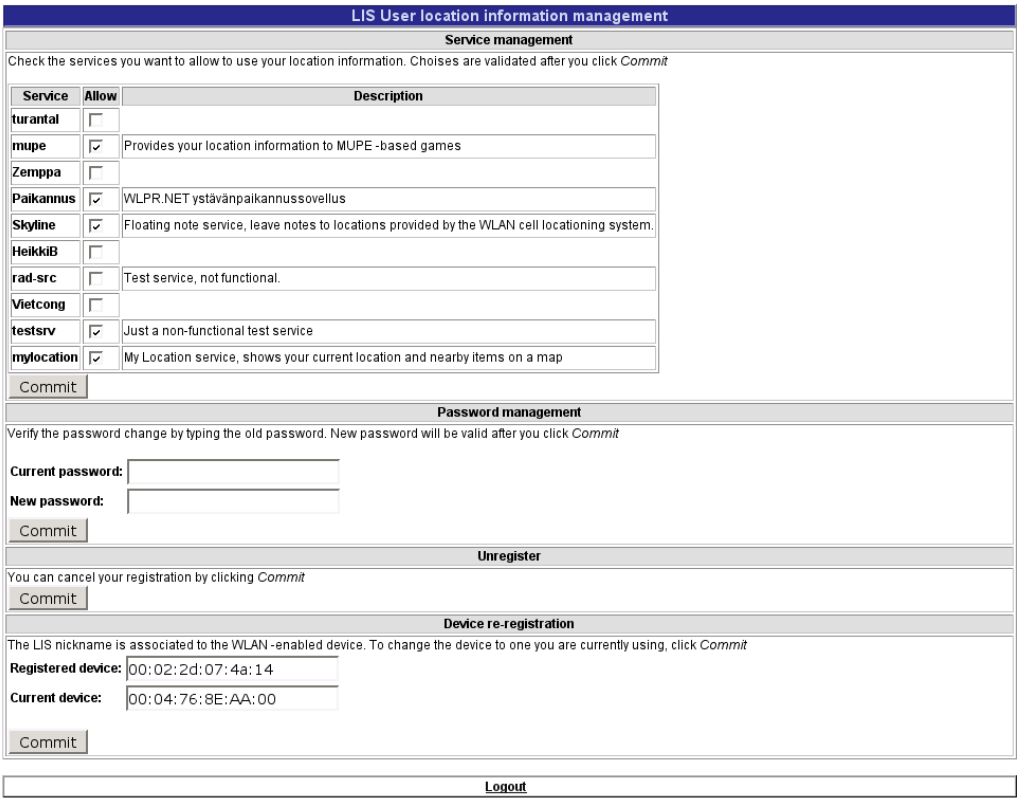
SOAP -rajapinnan ilmoitusmetodeja, tulee myös täyttää palvelun attribuuteissa (kuvassa 3.3 attributes -otsikon alla) olevat proxy sekä XML -nimiavaruus. Proxy kertoo minne ilmoitus tehdään (HTTP -palvelin) ja XML -nimiavaruutta käytetään muodostettavissa SOAP -viesteissä nimiavaruutena.



Kuva 3.3: Palveluntarjoajan käyttöliittymä

Kuvassa 3.4 on esitetty käyttäjien käyttöliittymä paikkatiedon käytön hallintaan. Käyttäjät näkevät paikkatietoa hyödyntävät palvelut ja voivat palvelukohtaisesti sallia tai kieltää oman paikkatietonsa käytön. Käyttäjä saa tunnuksen rekisteröintivaiheessa salasanan, joka voidaan muuttaa käyttöliittymästä käsin. Käyttäjä voi myös poistaa tunnuksensa järjestelmästä käyttöliittymästä käsin. Mikäli käyttäjän WLAN -laite vaihtuu, voidaan se käydä päivittämässä käyttäjän profiiliin. Kirjauduttaessa LIS -palveluun, näytetään käyttäjällä sillä hetkellä käytössä olevan WLAN -laitteen MAC -osoite, joka voidaan tallentaa profii-

lissa olevan tilalle. Tällöin käyttäjällä voi olla kerrallaan käytössä vain yksi paikannettava laite.



Kuva 3.4: Käyttäjien käyttöliittymä

3.1.3 SOAP (Simple Object Access Protocol) -palvelurajapinta

Palvelurajapinnan kautta palvelut voivat tehdä pyyntöjä käyttäjien solusijainneista WLAN -verkossa. Palvelut voivat myös kysellä solusijainteihin liitettyjä tietoja (esim. solun tukiaseman sijaintikoordinaatit). SOAP -rajapinnan toiminnot on jaettavissa kahteen osaan:

- *Pyyntömetodit*
  Näitä metodeja käytetään paikkatiedon kyselyyn. Kuvaukset pyyntömetodeista parametreineen on esitetty taulukossa 3.1.

- *Ilmoitusmetodit (triggerit)*
  Näillä metodeilla voidaan seurata yksittäisten käyttäjien liikkeitä. Kun käyttäjän paikassa tapahtuu muutos, ilmoittaa LIS -palvelu tästä uuden sijainnin tekemällä SOAP

-kutsun palveluun. Ilmoituksien käsittelyyn liittyvät metodit on esitetty taulukossa 3.2.

Taulukko 3.1: LIS -palvelurajapinnan pyyntömetodit

Metodi	**Kuvaus**
GetUserLocation	Hakee annetun käyttäjän (tai MAC -osoitteen) viimeisimmän sijainnin
	Parametrit
	target: Haun kohde *caller*: Haun tekijä (ei käytetty) *service*: Palvelu joka tekee haun *password*: Palvelun salasana *ttype*: Kohteen tyyppi (käyttäjä, mac) **paluuarvo**: GetUserLocationResponse -elementti jossa sekä solu (*location_id*), että aikaleima (*timestamp*) jolloin havainto tehtiin elementteinä
Metodi	**Kuvaus**
GetLocationInfo	Hakee tietoja annetusta solutunnisteesta
	Parametrit
	location_id: Solutunniste josta tietoa pyydetään *location_type*: Haetun tiedon tyyppi *service*: Palvelu joka tekee haun *password*: Palvelun salasana **paluuarvo**: Pyydetty tieto tietorakenteessa, joka sisältää pyyden tiedon tyyppiset kentät
Metodi	**Kuvaus**
GetLocationList	Hakee listan kaikista paikannusjärjestelmän soluista
	Parametrit
	service: Palvelu joka tekee haun *password*: Palvelun salasana **paluuarvo**: Taulukko solutunnisteista

Pyyntömetodeissa voi kohteena käyttää käyttäjän nimeä tai MAC -osoitetta. Käyttäjän nimeä käyttämällä voidaan yksittäinen käyttäjä identifioida, mutta MAC -osoitteella identifioidaan vain yksittäinen laite. Riippuen palvelusta, voi olla mielekästä käyttää MAC -osoitetta. Koska ei voida olla varmoja käyttösäännöistä, jotka yksittäistä MAC -osoitetta koskevat, käydään LIS -palvelussa läpi mahdolliset laitteen käyttäjät. Kyselyssä pyydetty tieto annetaan vain, mikäli kaikki mahdolliset laitteen käyttäjät antavat tähän suostumuksen. On siis mahdollista, että yhdellä laitteella on useampi käyttäjä, joita koskevat

ristiriitaiset käyttösäännöt.

Mikäli pyynnön kohteena olevalla MAC -osoitteella identifoidulla laitteella ei ole käyttäjää, sovelletaan oletussääntöjä. LIS -palvelussa voidaan määrittää palveluita, joille paikkatieto voidaan antaa joka tapauksessa. Tällainen palvelu voi olla esimerkiksi hätäpalvelu, joka hätätilanteessa paikantaa käyttäjän ja pyrkii auttamaan viranomaisia.

Pyyntömetodien käyttämät solutunnisteet on nimetty siten, että tunnisteen alkuosa ilmaisee organisaation ja lopuosa sijainnin organisaation tiloissa. Esimerkiksi LTY:n sisätiloissa olevan solun nimi voi olla vaikkapa *LTY_ 6606*, jossa 6606 ilmaisee tukiasemaa lähinnä sijaitsevan huoneen numeron. Solutunniste on sinänsä yleinen ja yksilöi vain tukiaseman. Sijaintitiedon esittäminen pelkällä solutunnisteella on vaikeaa, joten tunnisteisiin voidaan liittää lisätietoa.

Solutunnisteeseen liittyväksi tietotyypeiksi on tällä hetkellä määritetty kuvaus sekä solun tukiaseman sijaintikoordinaatit. Sijaintikoordinaatteina on käytetty Lappeenrannan kaupungin koordinaatistoa. Tukiaseman sijaintikoordinaatit ovat siten käytettävissä mm. Lappeenrannan kaupungin karttapalvelussa[2]. Solua esittävä kuvaus pyrkii esittämään tukiaseman ympäristön tekstipohjaisesti.

[2] `http://kartta.lappeenranta.fi`

Taulukko 3.2: LIS -palvelurajapinnan ilmoitusmetodit

Metodi	**Kuvaus**
SetTrigger	Asettaa "triggerin" määritellylle käyttäjälle. Muutokset käyttäjän liikkeissä raportoidaan takaisin triggerin asettaneelle palvelulle.
	Parametrit
	target: Käyttäjä jota halutaan seurata *service*: Palvelu joka tekee haun *password*: Palvelun salasana *uri*: Nimiavaruus josta josta ilmoituksen tekemiseen käytettävää *FireTrigger* -metodia kutsutaan *proxy*: URL osoite josta ilmoitusmetodia kutsutaan **paluuarvo**: Triggerin tunnistenumero
Metodi	**Kuvaus**
DeleteTrigger	Poistaa aiemmin asetetun "triggerin"
	Parametrit
	trigger_id: Seuranta-"triggerin" tunnistenumero *service*: Palvelu joka tekee haun *password*: Palvelun salasana **paluuarvo**: Poistetun triggerin tunnistenumero
Metodi	**Kuvaus**
FireTrigger	Ilmoitusmetodi, jolla palvelulle ilmoitetaan seuratun käyttäjän liikkeistä
	Parametrit
	trigger_id: Seuranta-"triggerin" tunnistenumero *location_id*: Uusi solusijainti, jonne käyttäjä on liikkunut **paluuarvo**: Ei käytössä

Ilmoitusmetodien päätarkoitus on mahdollistaa tietyn käyttäjän seuraaminen. Ilmoitusmetodien käytöllä on tarkoitus antaa palveluille mahdollisuus toimia ilmoituspohjaisesti sen sijaan, että käyttäjien liikkeitä kyseltäisiin jatkuvasti. Jotta ilmoituksia paikan vaihdoksesta voitaisiin vastaanottaa, tulee palvelun ensin rekisteröidä halukkuus vastaanottaa ilmoituksia. Tämä tapahtuu *SetTrigger* -metodilla, jolla määritetään käyttäjä, jota halutaan seurata. Kutsun jälkeen palvelu saa tunnistenumeron, jolla palvelu voi identifioida asettamansa ilmoituspyynnöt.

Kun määritetyn käyttäjän paikkatiedossa tapahtuu muutos, ilmoitetaan tästä palvelulle SOAP -rajapinnan avulla. Ilmoituksen vastaanottaminen tapahtuu *FireTrigger* -metodilla, jonka palvelun tulee toteuttaa. Tällä hetkellä kuljetuskerroksena käytetään HTTP -protokollaa, joten *FireTrigger* -metodin toteutuksen tulee olla HTTP -protokollaan sidottu. Kun paikan muutoksesta ilmoitetaan, LIS kutsuu paikkatietoisen palvelun toteuttamaa *FireTrigger* -metodia. Kutsu tapahtuu muodostamalla SOAP -protokollan mukainen XML -dokumentti ja lähettämällä se HTTP -protokollan avulla palvelun määrittämälle HTTP -palvelimelle.

3.1.4 Tietokantarajapinta

LIS -palvelu kommunikoi paikkatietokantana käytetyn PostgreSQL -tietokannan kanssa Perl DBI -moduulin [29] avulla. DBI -moduuli on standardi tietokantakirjastorajapinta Perl -ohjelmointikielelle. DBI -moduuli määrittää metodit, muuttujat ja käytännöt, jotka muodostavat yhtenäisen rajapinnan käytetystä tietokannasta riippumatta. DBI -kirjaston metodeja käytetään tietokannan tietojen hakuun ja päivitykseen.

Tietokantarajapinnassa on toteutettu myös toiminnallisuutta PostgreSQL -tietokannan sisälle ilmoitusrajapinnan käyttötarkoituksiin. Ilmoitusrajapinnassa käytetään PostgreSQL -tietokannan mahdollisuutta ohjelmallisesti täydentää tietokannan toimintoja. Tietokannassa on mahdollista laajentaa tietokannan toiminnallisuutta luomalla funktioita ([30], s.290-291). PostgreSQL tukee neljän tyyppisiä funktioita ([31], kappale 12.1):

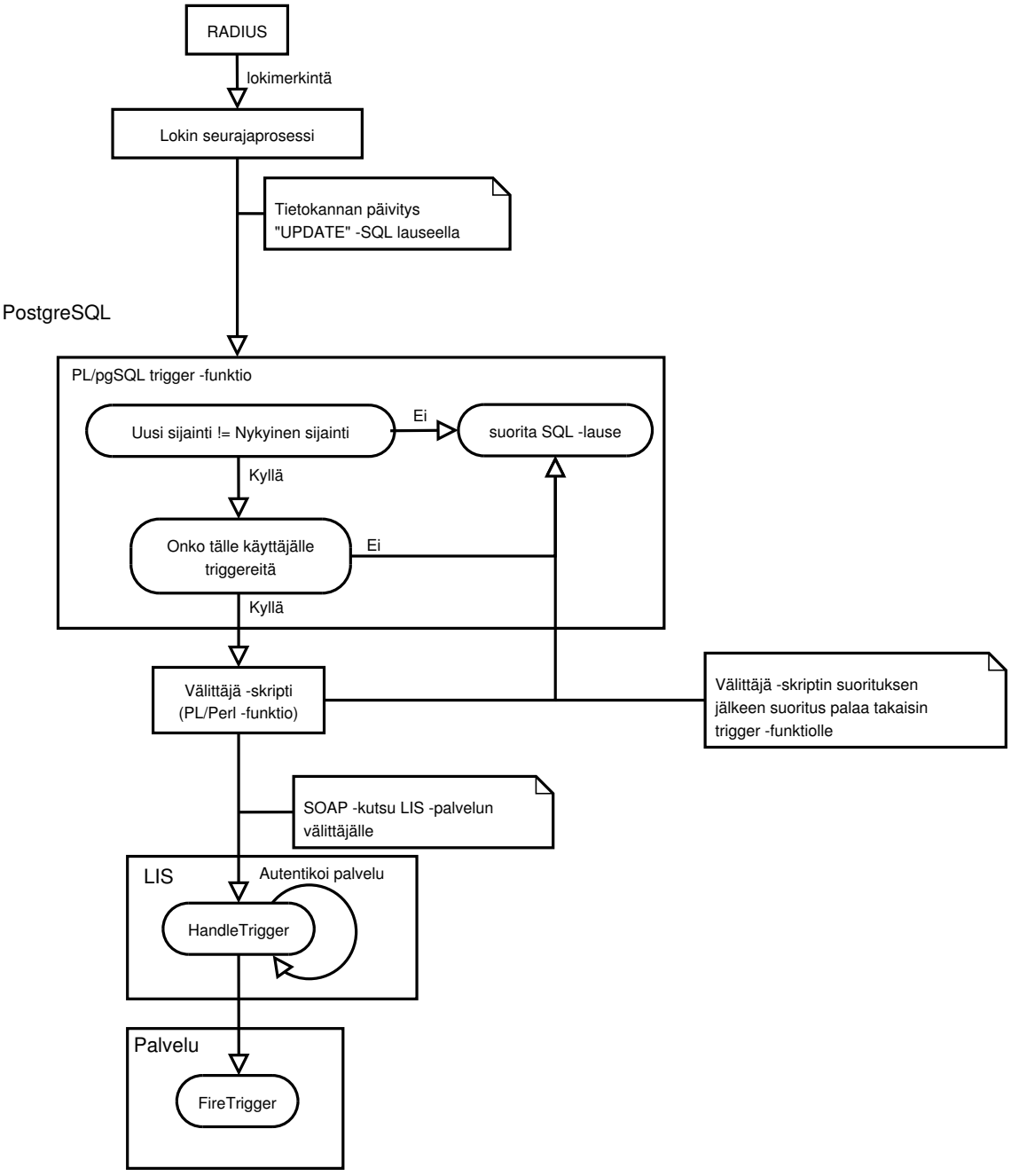
- Kyselykieliset funktiot (kirjoitettu SQL -kielellä)
- Proseduraaliset funktiot (funktiot jotka on kirjoitettu esimerkiksi PL/pgSQL tai PL/Perl -kielillä)
- Sisäiset funktiot
- C -kielellä kirjoitetut funktiot

Ilmoitusmetodien toteutukseen käytettiin proseduraalisia funktioita[3]. Proseduraalisissa funktioissa PostgreSQL -tietokanta itse ei tulkitse funktion lähdekieltä, vaan funktion lähdekielen tulkinta, syntaksin analysointi, suoritus ja muut operaatiot annetaan funktion lähdekielen mukaisen käsittelijän tehtäväksi. Tämä antaa mahdollisuuden käyttää esimerkiksi Perl -kieltä tietokannasta käsin SQL -operaatioiden yhteydessä.

Funktiolaajennuksien lisäksi käytettiin triggereitä. Triggeri on funktio, joka suoritetaan ennen tai jälkeen jonkun muun taululle tehdyn tapahtuman ([30], s.278-279). Triggerifunktio voidaan määrittää laukaistavaksi esimerkiksi tietokannan tietyn taulun päivittämisen (UPDATE SQL -lause) yhteydessä. Triggerifunktion avulla voidaan tällä tavoin esimerkiksi pitää lokia tietokannan tapahtumista. Triggerifunktio voidaan toteuttaa PostgreSQL -tietokannan tukemilla funktionaalisilla kielillä, poislukien SQL -kieli.

LIS -palvelun ilmoitusrajapinnan toteutuksessa hyödynnettiin PL/pgSQL ja PL/Perl proseduraalisten kielten yhdistelmää. Kuvassa 3.5 on esitetty ilmoitusmekanismin toiminta, sekä toiminnan toteutukseen käytetyt proseduraaliset funktiot. Liitteessä 1 on esitetty PL/pgSQL -trigger funktion toiminta ja liitteessä 2 PL/Perl -välittäjäskriptin toiminta.

[3] Tietokantaan tallennetuista funktioista käytetään myös nimitystä “stored procedures”

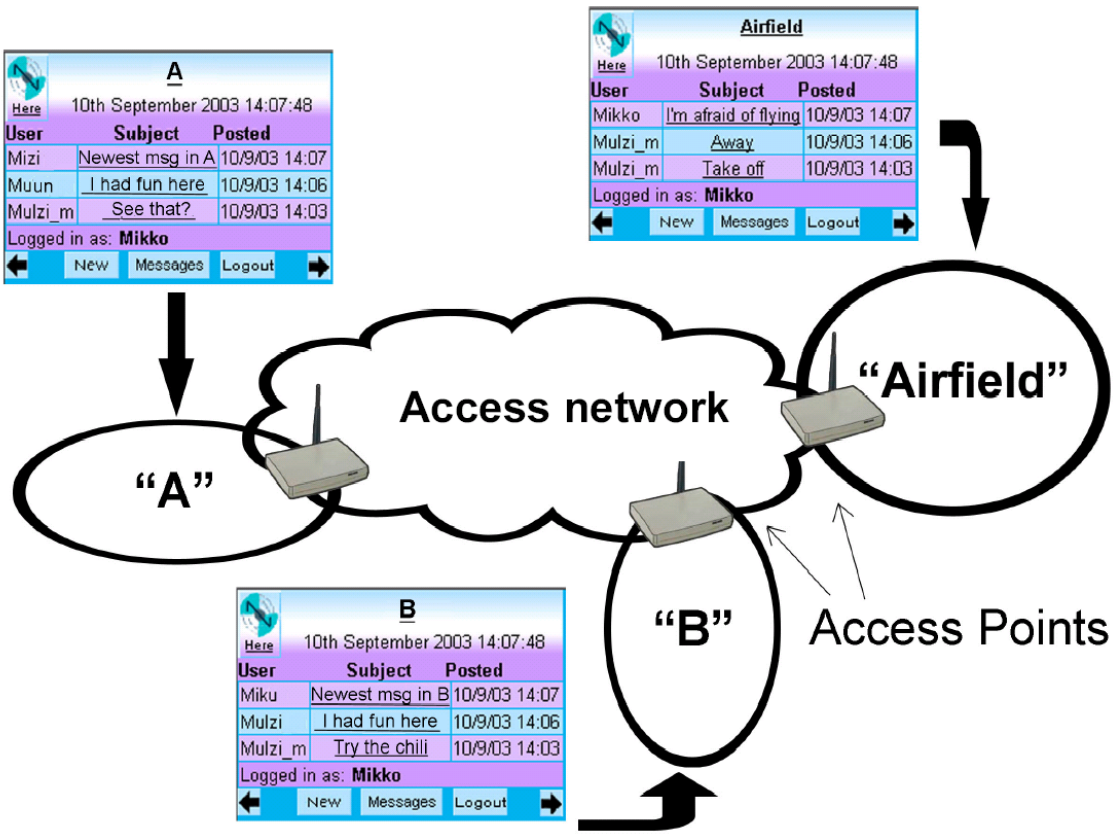


Kuva 3.5: Triggerien käyttö PostgreSQL -tietokannassa.

Kun paikkatietokantaan syötetään WLAN -laitteen sijaintitiedon päivitys UPDATE -lauseella, laukaistaan UPDATE -lauseeseen yhdistetty PL/pgSQL trigger -funktio. Funktiossa vertaillaan ensin, eroaako päivitettäväksi tarkoitettu uusi sijainti vanhasta sijainnista. Mikäli sijainti eroaa, tarkistetaan onko kyseiselle laitteelle triggereitä (tässä yhteydessä triggeri tarkoittaa paikkatietoisen palvelun asettamaa seurantapyyntöä). Mikäli laiteelle löytyy triggeri, suoritetaan PL/Perl -funktio (2), joka vuorostaan välittää ilmoituspyynnön SOAP -kutsuna LIS -palvelulla. Kutsun parametreina välitetään uusi sijainti sekä triggerin tunniste. LIS vuorostaan kutsuu paikkatietoista palvelua, jolle ilmoitus välitetään kutsumalla palvelurajapinnassa määritettyä *FireTrigger* -metodia. *FireTrigger* -metodi välittää ilmoitusta haluavalle palvelulle uuden sijainnin sekä triggerin tunnisteen. Tällä menettelyllä on mahdollista seurata paikkatietopäivityksiä sitä mukaa, kun niitä tietokantaan päivitetään. Järjestelmän huono puoli on lisäprosessointi, jota LIS -palveluun sekä tietokantaan aiheutuu.

3.1.5 Floating Note -viestipalvelu

LIS -palvelua käytettiin toteuttaamaan erikoistyönä WLPR.NET -verkon käyttöön paikkatietoinen viestintäpalvelu, Floating Note [32]. Floating Note -palvelu on paikkatietoinen viestintäpalvelu. Solupaikkatietoa käytetään luomaan sijainteihin "keskustelualueita". Keskustelualueiden aluekohtaisessa jaossa pyritään keskustelua ohjaamaan sijaintiin liittyviin aiheisiin. Esimerkiksi lentokentällä keskusteluaiheet voivat liittyä lentokentän toimintaan. Kuvassa 3.6 on esitetty esimerkkikuva "Floating Note" -viesteistä. Jokainen solusijainti muodostaa oman keskustelualueensa.



Kuva 3.6: Esimerkki paikkasidonnaisista Floating Note -viesteistä [32]

Käyttäjät luovat tilin Floating Note -palveluun, jonka jälkeen he voivat kirjoittaa viestejä. Tilin luonnin yhteydessä käyttäjä antaa LIS -palveluun luoman identiteetin, jota käytetään paikannuspyyntöjen tekemiseen. Kun käyttäjä kirjautuu palveluun, näytetään hänelle nykyisessä sijainnissa olevat viestit. Viestit koostuvat otsikosta ja tekstistä, johon voidaan myös liittää kuvia. Floating Note -järjestelmä pitää kirjaa mahdollisista solusijainneista ja solusijaintien kuvauksia voidaan myös muuttaa ylläpidon toimesta. Järjestelmän ylläpitäjä huolehtii myös viestien ja kuvien hallinnoinnista sekä niiden mahdollisesta poistamisesta.

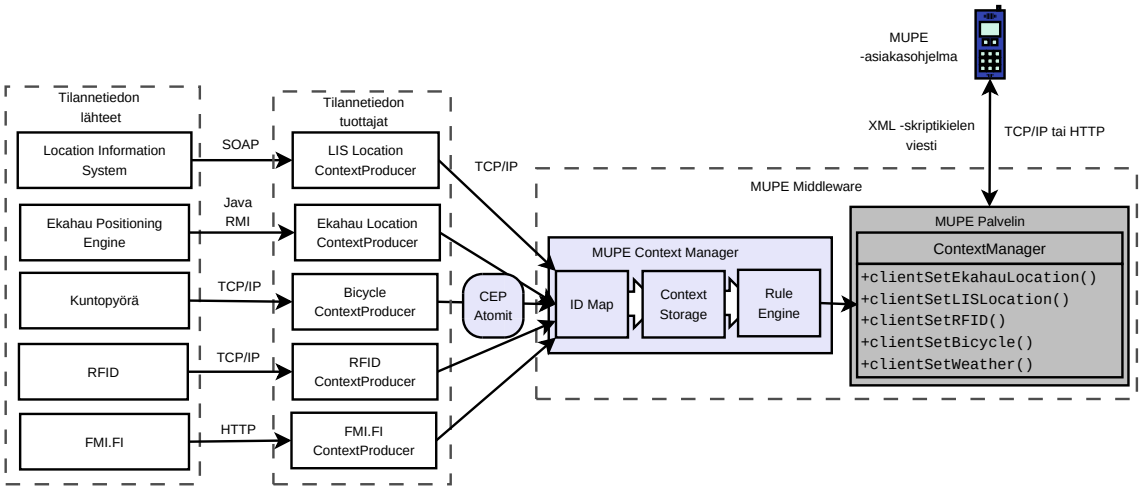
Eri sijainteihin liitettyjä viestejä voidaan selata, joten viestintä ei ole pelkästään sidottu yhteen sijaintiin. Palvelun toimintaideana on kuitenkin luoda keskustelua, joka liittyisi tiettyyn sijaintiin.

3.2 MUPE -alustan sovellukset

Tilanne- ja paikkatiedon käyttöä sovellettiin MUPE -alustalla tilanne- ja paikkatietoisten sovelusten kehittämiseen. Sovellusten kehittämiseksi toteutettiin MUPE -alustan tilannetietorajapinnalle, CEP (Context Exchange Protocol), tilannetiedon tuottajakomponentteja (*Context Producer*). Tilannetiedon lähteinä toimivat WLPR.NET WLAN -verkko, ilmatieteen laitoksen sääpalvelu, Tunturi recumbent -kuntopyörä sekä RFID (Radio Frequency Identification) -tunnisteiden lukija. WLAN -verkosta kerättiin paikkatietoa kahdella eri järjestelmällä. Nämä järjestelmät olivat kappaleessa 3.1 esitetty LIS -järjestelmä sekä Ekahau -yrityksen tuottaman EPE (Ekahau Positioning Engine) 2.0. LIS -järjestelmän avulla saatiin kerättyä solupohjaista paikkatietoa. Ekahau -järjestelmän avulla tuotettiin LTY:n sisätiloista, erityisesti tietoliikennetekniikan laitoksen läheisistä tiloista, rakennuksen pohjapiirustuksiin pohjaavaa paikkatietoa. Toteutettuja tilannetiedon tuottajia käytettiin hyväksi MUPE -sovellusten kehitykseen sekä LTY:ssä järjestetyn Sovelluskehityksen erikoiskurssin aikana sekä Tribal Exchange -peliprojektissa.

3.2.1 Tilannetiedon tuottajat

Kuvassa 3.7 on esitetty toteutus miten tilannetietoa käsitellään MUPE -sovelluksen sisällä. *MUPE Server* -komponentin sisältämä *Context Manager* -objekti sisältää metodit, jotka käsittelevät vastaanotetun tilannetiedon (esim. *clientSetEkahauLocation()*). Haluttu tilannetieto välitetään näiden metodien parametreisssa. *MUPE Context Manager* -komponettiin toteutettiin jokaista tilannetiedon tuottajaa varten XML -skriptit, joiden avulla MUPE -sovelluksen *Context Manager* -objektin metodeja kutsutaan. Esimerkki näistä skripteistä on esitetty liitteessä 4. Liitteessä olevaa skriptiä käytettiin välittämään LIS -palvelurajapinnalta pyydetty uusi paikkatietonäyte MUPE -sovelluksen metodille *clientSetLocation*.



Kuva 3.7: Tilannetiedon tuottajien yleiskuvaus

Säätietoa haettiin Ilmatieteen laitoksen tarjoamalta WWW -sivustolta[4], jolta löytyy paikkakuntakohtaisesti säähaintotiedot. Tiedot päivittyvät tunnin välein. Säätiedon tuottaja hakee määräajoin HTTP -protokollan avulla säähavaintosivuston ja parsii tästä sivusta säähavainnot. Kun havainnot on parsittu, ne lähetetään yhteyden ottaneille MUPE -sovelluksille CEP -protokollan mukaisina XML -viesteinä (CEP atomit). Säähavaintoja voidaan lähettää MUPE -palvelimille tiheämmin kuin tunnin välein, mutta tällöin havaintotiedot pysyvät samoina tunnin sisällä.

LTY:n tietoliikenteen laitokselle oli hankittu PTD (Personal Trusted Device) -tutkimusprojektiin Tunturi recumbent -kuntopyörä. Kuntopyörästä voidaan lukea sekä pyörän nopeus että pyörää sillä hetkellä polkevan käyttäjän syke. Nämä tiedot luetaan sarjaporttiliitännän avulla käyttäen Tunturin määrittämää protokollaa. Harjoitustiedon (nopeus ja syke) lukua varten oli toteutettu ohjelmistokomponentti, joka luki harjoitustietoa kuntopyörästä ja vastaanotti TCP -yhteyksiä. TCP -yhteyksien kautta harjoitustiedon välitykseen määriteltiin yksinkertainen protokolla, jolla voitiin kysellä pyörän sen hetkistä nopeutta. Harjoitustieto muutettiin tämän jälkeen CEP -protokollan mukaiseksi tilannetiedon tuottajakomponentissa. Tilannetiedon tuottajakomponentti vuorostaan palveli MUPE -sovelluksia lähettämällä nykyisen nopeuden ja sykkeen CEP -protokollan avulla.

RFID -tunnisteiden lukua varten LTY:n tietoliikenteen laitokselle oli hankittu useita RFID -tunnisteita (henkilökorttipohjia sekä avaimenperätunnisteita). Tunnisteiden lukuun käytettiin RFID -lukijaa, jossa on sarjaporttiliitäntä, jonka kautta lukija ilmoittaa havainnon

[4] `http://www.fmi.fi`

lukijan editse (muutaman senttimetrin etäisyydeltä) viedystä RFID -tunnisteesta. Tässä havainnossa kerrotaan mm. 16 tavun mittainen RFID -tunnisteen yksilöivä tunnistenumero. Tunnisteiden luku toteutettiin samantyyppisesti kuin harjoitustiedon luku kuntopyörästä. Ohjelmistokomponentin avulla tulkittiin RFID -lukijan lähettämiä havaintoja sarjaporttiliitännän kautta. Komponentti vastaanotti myös TCP -yhteyksiä, joihin havaintotiedot välitettiin. Toteutetun tilannetiedon tuottajakomponentin tehtäväksi jäi muodostaa TCP -yhteys RFID -lukijaa käsittelevään komponenttiin ja lukea saatuja RFID -havaintotietoja muodostetusta TCP -yhteydestä. Havaintotiedot muunnettiin edelleen CEP -protokollan atomeiksi ja välitettiin MUPE -sovelluksille.

Paikkatiedon välitykseen MUPE -sovelluksille käytettiin kahta järjestelmää. LIS -solupaikannusrajapinnalle toteutettiin tilannetiedon tuottaja joka hyödynsi LIS -järjestelmän pyyntömetodeja. Tilannetiedon tuottajakomponentille määritettiin seurattavien WLAN -päätelaitteiden laiteosoitteet. Tämän jälkeen tilannetiedon tuottajakomponentti kyseli määrätyin väliajoin määritettyjen päätelaitteiden sijaintia LIS -järjestelmältä. LIS -palvelun kanssa kommunikointiin käytettiin Apache Axis -projektin[5] SOAP -toteutusta. Listauksessa 3.1 on esitetty esimerkki LIS -järjestelmän tuottamasta paikkatiedosta CEP -protokollan mukaiseksi XML -viestiksi muotoiltuna. CEP -viestissä kerrotaan laiteosoitteella *00:08:02:-F6:01:7F* määritetyn käyttäjän olevan solusijainnissa *LTY_6609*. Lisäksi CEP -viestiin on sisällytetty havainnon aikaleima sekä kuvaus solusijainnista.

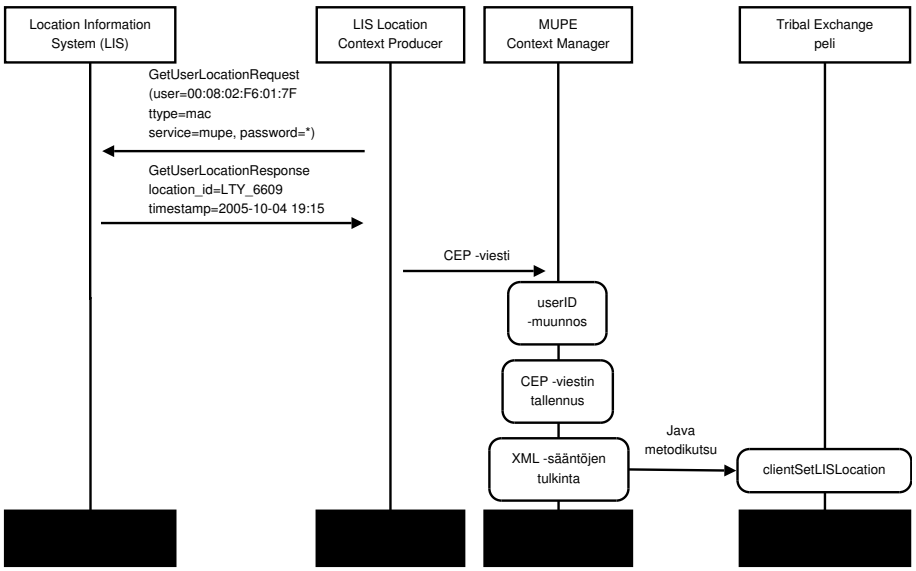
```
<atom name='LISLocation' timestamp='2005-9-5 1:23:26,00 +1'
      source='http://gamesrv.wlpr.net:1234' userId='00:08:02:F6:01:7F'>
  <string name='timestamp'>2005-10-04 19:15:35.081464+03
  </string>
  <string name='location'>LTY_6609
  </string>
  <string name='description'>LUT, TITE building, 6th floor (Comlab)
  </string>
</atom>
```

Listaus 3.1: Solupaikkatieto CEP -protokollan XML -muodossa.

Kuvassa 3.8 on esitetty viestikaavio paikkatiedon välityksestä LIS -solupaikannusjärjestelmästä tilannetiedon tuottajan avulla MUPE -sovellukseen. Kaaviosta nähdään tilannetiedon tuottajakomponentin (LIS Location Context Producer) tekemä pyyntö LIS -järjestelmään parameterineen. Myös vastaus parameterineen on esitetty. Tilannetiedon

[5] http://ws.apache.org/axis/

tuottajakomponentille voitiin määrittää väliajat, jolloin kyselyjä annettujen laitteiden sijainneista tehtiin LIS -järjestelmään. Laitteet puolestaan määritettiin XML -konfiguraatiotiedostossa.



Kuva 3.8: Paikkatiedon välitysketju LIS -järjestelmästä

EPE (Ekahau Positioning Engine) 2.0 -järjestelmän avulla toteutettiin tarkempaa, koordinaattipohjaista paikkatietoa tuottava tilannetiedon tuottaja. Paikkatiedon kartoitusalustana käytettiin LTY:n tieto- ja sähkötekniikan rakennuksen kerroksia 2-7. Tilannetiedon tuottajan toteutuksessa käytettiin EPE:n tarjoamaa Java RMI -rajapintaa. Tilannetieton tuottaja toimi client -sovelluksena EPE -järjestelmään. EPE -järjestelmä tuottaa kahden sekunnin välein uusia paikkatietohavaintoja. Näistä havainnoista kerättiin mm. koordinaatit, kerros, todennäköisin looginen sijaintialue sekä todennäköisyys kyseisellä loogisella alueella sijaitsemiseen. Tiedot muokattiin CEP -atomeiksi ja päivitettiin määrätyin väliajoin MUPE -sovelluksille. EPE -järjestelmää käyttävä tilannetiedon tuottaja poikkesi LIS -järjestelmää käyttävästä siten, että paikkatietoa ei erikseen pyydetty lähettämällä pyyntö vaan sitä vastaanotettiin jatkuvana virtana.

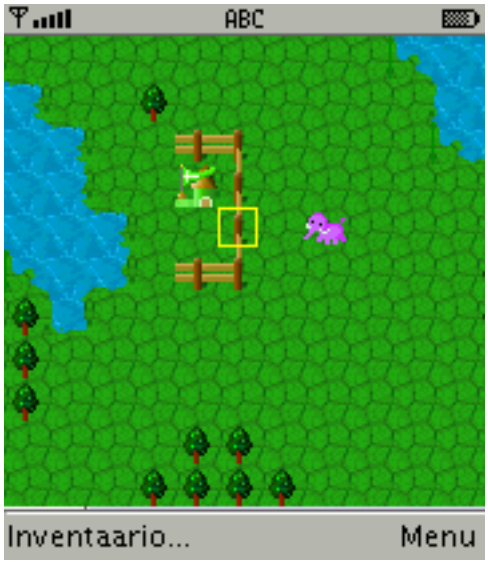
3.2.2 Tribal Exchange -peli

Edellä kuvattuja MUPE -alustalle toteutettuja tilannetiedon tuottajakomponentteja hyödynnettiin Tribal Exchange -peliprojektissa. Tribal Exchange (Trix) -peli toteutettiin AMPERS -projektissa yhdessä Lapin Yliopiston (LaY) taiteiden tiedekunnan kanssa. Osuu-

teni projektissa koostui suunnittelusta sekä erinäisten osien toteutuksesta. Tärkein toteutukseen liittyvä komponentti omalta osaltani oli tilannetiedon tuominen Trix -peliin. Tähän käytettiin sekä edellä kuvattuja säätiedon että solupojaisen WLAN -paikkatiedon (LIS) tuottavia tilannetiedon tuottajakomponentteja. Peliprojektin määrittely aloitettiin lokakuussa 2004. Määrittelyyn sisältyi mm. osapuolien tutustuttaminen MUPE -alustaan. Suunnittelua tehtiin marraskuun 2004 ja tammikuun 2005 välisenä aikana, jolloin suurin osa pelin eri toiminnoista lyötiin lukkoon. Suunnittelussa käytettiin apuna AMPERS -projektin käyttöön asennettua MoinMoin Wiki -palvelua[6], johon kerättiin eri suunnittelupalaverien tulokset. Kevät 2005 käytettiin pitkälti pelin prototyypin rakentamiseen. Alkukesän 2005 aikana peliprototyyppiin lisättiin suurin osa suunnitelluista toiminnoista. Peliprojektin testauksesta ei vielä tällä hetkellä ole täyttä varmuutta. Ilmeisesti Lapin Yliopistolla on kuitenkin tarkoitus käyttää prototyyppiä jatkokehittelyyn.

Pelin ideana on yhdistää sekä fyysistä peliympäristöä että pelin virtuaalimaailmaa toisiinsa. Pelaajat jaetaan heimoihin. Pelimaailma koostuu pelikartasta, jossa jokaisella heimolla on taloja. Yksi taloista on päämaja ja muut ovat heimolaisten asuinrakennuksia. Pelin ideana on suojella asuinrakennuksia erilaisilta luonnonmullistuksilta. Suojeleminen tapahtuu rakentamalla muuria talojen ympärille. Muurin rakennusidea on johdettu vuonna 1991 julkaistusta Rampart -pelistä. Peli on jaettu jaksoihin, joiden aikana pelaajien tulee suojella ilman muuria olevat rakennukset. Jaksojen päättyessä tarkistetaan ovatko heimojen asuinrakennukset ympäröity. Jos onnistuttiin, heimo saa uuden asuinrakennuksen pelikartalle. Mikäli olemassaolevia rakennuksia ei ole ympäröity, poistetaan ne pelikartalta. Kuvassa 3.9 on esitettu ruudunkaappaus pelitilanteesta. Kuvan tilanteessa MUPE -asiakasohjelmaa ajetaan J2ME (Java 2 Micro Edition) Wireless Toolkit MIDP -emulaattorin avulla. MUPE -asiakasohjelma on kytkeytyneenä Trix -pelipalvelimelle ja kuvan tilanteessa vihreän heimon päämajasta osa on ympäröity muurilla.

[6] `http://moinmoin.wikiwikiweb.de/`



Kuva 3.9: Ruutukaappaus Tribal Exchange -pelistä.

Jotta muuria voisi rakentaa, pelaajat keräävät pelin peluuympäristöstä koodeja. Koodit ovat vapaaluontoisia merkkijonoja, jotka voivat esiintyä esimerkiksi paperilapuilla. Koodit vastaavat pelimaailmassa hyödykkeitä; joko muurin rakennuspaloja tai käyttöesineitä, joilla voidaan raivata pelikartalla rakennustilaa muurille. Pelimaailman maastossa voidaan rakentaa muuria vain tasaiselle maalle. Esteinä toimivat vesi, kivet ja metsä. Käyttöesineitä on kolmea tyyppiä. Hakulla voidaan hakata kivet pois muurin rakennusalueelta. Sahalla voidaan kaataa metsää pois muurin rakennusalueelta. Dynamiitilla voidaan tuhota sekä muuria, metsää että kiveä. Jokainen käyttöesine voidaan käyttää vain kerran.

Koodeilla saatavat muuripalat kuuluvat jollekin pelin heimoista ja vaihtelevat muotonsa mukaan (L, Z, I, O ja C -kirjainten mukaiset muodot) sekä rakennusaineen mukaan. Muurityyppejä ovat olki, puu, jää tai kivi. Muurityyppien kestävyys vaihtelee heikoimmasta (olki) vahvimpaan (kivi). Muurin omistajaheimo määritetään palan värin mukaan. Mikäli pelaajan koodilla hankkima muuripala on hänen oman heimonsa pala, muuttuu se pelikartalle asettamisen yhteydessä kivimuuriksi. Mikäli muuripala on jonkin toisen heimon omistama, muuttuu se satunnaisesti joksikin muuksi heikommaksi muurityypiksi. Samanmuotoisia muuripaloja voidaan yhdistellä keskenään. Mikäli kaksi samanväristä (saman vieraan heimon omistuksessa olevaa) ja samanmuotoista palaa yhdistetään, saadaan tulokseksi yksi oman heimon omistuksessa oleva pala. Erivärisiä paloja yhdistellessä on 50% todennäköisyys saada tulokseksi oman heimon pala ja 50% todennäköisyydellä vieraan heimon pala. Pelissä voidaan myös tehdä vaihtokauppaa muiden pelaajien kanssa. Muurin-

palojen omistussuhteiden onkin tarkoitus edistää pelaajien välistä interaktiota, sillä väärän heimon palan voi laittaa huutokaupassa myytäväksi ja täten hankkia lisää oman heimon paloja. Lisäksi peli sisältää keskustelualueen (heimokohtaisen sekä julkisen), jonka kautta pelaajat voivat viestiä toisten pelaajien tai heimon jäsentensä kesken. Trix -pelin toiminnot on jaettu neljään toimintamoodiin jotka on esitetty taulukossa 3.3.

Taulukko 3.3: Tribal Exchange -pelin toimintamoodit

Pelimoodi	**Kuvaus**
Inventaaariomoodi	Inventaariossa pelaajalle esitetään hänen hallussaan olevat käyttöesineet sekä muuripalat. Inventaarioon päätyvät myös huutokaupassa sekä koodeilla saadut esineet. Inventaariossa ollessa voidan myös etsiä koodeja käyttäjän nykyisestä sijainnista.
Rakennusmoodi	Rakennusmoodissa pelaajat voivat tarkastella 9*9 pikselin kokoisista ruuduista koostuvaa pelikarttaa. Päätelaitteen ruudulla esitetään aina niin suuri osa pelikarttaa kuin mahdollista. Inventaariosta voidaan ottaa käyttöön muuri- tai hyödyke-esineitä ja käyttää niitä rakennusmoodissa. Muuripaloja voidaan pyöritellä myötä/vastapäivään ja asettaa vapaalle kartta-alueelle. Hyödykkeitä (dynamiitti, hakku, saha) käytetään valitsemalla esine inventaariosta ja viemällä se rakennusmoodissa käytettävän ruudun ylle. Esimerkiksi sahan käyttäminen metsää sisältävässä ruudussa johtaa metsän poistumiseen.
Vaihtokaupamoodi	Vaihtokauppamoodissa pelaajat voivat jättää vaihtokauppatarjouksia ja vastata muiden pelaajien jättämiin tarjouksiin. Mikäli pelaaja hyväksyy vaihtokaupan, siirretään vaihtokaupassa vaihdetut esineet vaihtokaupan osapuolien inventaarioihin.
Keskustelumoodi	Keskustelumoodissa pelaajat voivat viestiä keskenään. Keskustelu on jaettu kahteen alueeseen, pelaajan oman heimon keskiseen ja yleiseen keskustelualueeseen. Pelaajat voivat vapaasti jättää ja lukea viestejä.

Pelialueella on maastoesteiden lisäksi erilaisia tapahtumia, jotka haittaavat muurin rakentamista. Näitä ovat metsäpalot ja kulkevat mammutit. Metsäpaloja voi syttyä satunnaisesti ja ne tuhoavat levitessään olkimuureja. Mammutit taas liikkuvat satunnaisesti ja syövät tielleen osuvaa olkimuuria. Mammutit toimivat myös esteinä, jolloin ne vaikeuttavat muuripalojen asettelua.

Koodeja voidaan etsiä myös paikkatiedon perusteella. Koodeille voidaan määrittää sijaintitieto WLPR.NET -verkossa käytetyn WLAN -solupaikannusjärjestelmän avulla. Paikkatieto tuodaan peliin LIS -rajapinnalle toteutetun tilannetiedon tuottajan kautta. Pelaajat voivat määrittää käyttämänsä WLAN -päätelaitteen laiteosoitteen (tai vaihtoehtoisesti ni-

men, mikäli käytetään MUPE:n oliotunnisteiden muunnostaulukoita) omassa pelaajaprofiilissaan. Tämän jälkeen pelaaja voi kysellä mahdollisia koodeja omasta, sen hetkisestä sijainnistaan. Koodeja voi olla samassa sijainnissa useita, mutta niistä voi käyttää vain yhden kerrallaan.

Käytön jälkeen koodi merkitään käytetyksi, joten muut pelaajat eivät voi enää hankkia hyödykkeitä tällä koodilla. Koodeja voidaan lisätä pelin ylläpidon toimesta pelimaailmaan lisää sitä mukaan, kun niitä käytetään. Paikkatiedon käytöllä saavutetaan se hyöty, että koodeja ei tarvitse jaella fyysisesti erillisille lapuille ympäri peliympäristöä. WLAN -paikkatiedon käyttö taas rajoittaa mahdollisia päätelaitteita tai näiden yhdistelmiä[7].

Trix -pelimaailma yhdistetään reaalimaailmaan myös säätiedon käytöllä. Säätietoa tuodaan peliin Ilmatieteen laitoksen[8] palvelua käyttävän tilannetiedon tuottajan kautta. Säähavainnoista käytetään hyväksi kolmea parametria: lämpötila, kosteus ja tuulen nopeus. Saataessa uusi säähavainto suoritetaan näille kolmelle eri parametrille annettujen ehtojen täyttyessä pelimaailmaan rakennetuille muuripaloille kestävyysmuutoksia. Kestävyysmuutokset on määritetty MUPE:n *Context Manager* -komponentin skriptien avulla (liite 3). Lämpötila vaikuttaa vain jäämuurinpaloihin, mutta kosteusprosentti vaikuttaa sekä olkeen että puuhun ja tuuli olkeen, puuhun ja jäähän. Näin ollen kestävin komponentti on kivi. Eri muurityyppeihin vaikuttavat sääparametrien muutokset on esitetty taulukossa 3.4. Eri muurityypeillä on ennalta määrätty kestävyysarvo, jonka loppuessa muuri sortuu. Pelaajien tulee paikata sortuneet palat uusilla muuripaloilla.

[7] Yhdistelmällä tarkoitetaan esimerkiksi WLAN -radion omaanvan kannettavan tietokoneen ja puhelimen yhdistelmää. Peliä voidaan pelata puhelimella, mutta koodeja voi etsiä WLAN -kannettavan avulla.

[8] `http://www.fmi.fi`

Taulukko 3.4: Tribal Exchange -pelin sääparametrien vaikutus muurityypeittäin

Lämpötila (°C)	**Vaikutus**
> +30	Jää: -4
+20 - +30	Jää: -3
+10 - +20	Jää: -2
0 - +10	Jää: -1
-10 - 0	ei vaikutusta
-20 - -10	Jää: +1
-30 - -20	Jää: +2
< -30	Jää: +3
Kosteus (%)	**Vaikutus**
0 - 60	ei vaikutusta
60 - 80	Olki: -1
80 - 100	Olki: -2, Puu: -1
Tuulen nopeus (m/s)	**Vaikutus**
0 - 2	ei vaikutusta
2 - 4	Olki: -1
4 - 8	Olki: -2
> 8	Olki: -3, Puu: -1, Jää: -1

Luku 4

Arviointi

Tässä kappaleessa esitetään arviointia ja yhteenvetoa edellisessä luvussa kehitetyistä palveluista, välittäjäkomponenteista ja sovelluksista. Erityisiä mittauksellisia tuloksia ei arviointia varten tehty, vaan arviointi pyritään tekemään rakenteellisesti. Tärkeimmät havainnot työn tuloksien osalta käydään läpi.

4.1 Käyttäjän ja päätelaitteen tunnistus

LIS -palvelussa käyttäjät tunnistetaan käyttäjätunnuksen avulla. Käyttäjätunnuksien avulla tietyn käyttäjän sijaintia voidaan kysellä. Tunnisteen luominen on tarvittavaa, sillä WLAN -verkoissa ei käyttäjän tunnistamiseksi ole olemassa yhtenäistä tapaa, kuten esimerkiksi GSM -verkoissa puhelinnumero. Käyttäjätunnuksella pyrittiin ottamaan myös huomioon, että paikannuspalvelun käyttäjä voi useasti olla eri henkilö kuin itse päätelaitteen (tai esim. GSM -liittymän) omistaja.

Käyttäjän identifiointi LIS -palvelussa tunnuksen perusteella luo tarpeen, että käyttäjän tunnusta tulee käyttää myös paikkatietoisessa palvelussa, jotta käyttäjä voidaan identifioida. Käyttäjän identifiointi voisi myös itsessään olla palvelu, jota LIS:n kaltainen paikkatietorajapinta voisi tarjota. Tämä huomattiin Floating Note -palvelua rakennettaessa, jolloin käytettiin myös käyttäjäprofiileja. Tämä johti tilanteeseen, jossa käyttäjällä oli identiteetti sekä Floating Note, että LIS -palvelussa. LIS -palvelun käyttäjätunnus tallennettiin osaksi Floating Note -palvelun käyttäjäprofiilia. Pelkästään LIS -palvelun käyttäjäidentiteetin käyttäminen olisi tässä tilanteessa ollut järkevää, sillä paikkatietokyselyjä tehdään kuitenkin kyseisen käyttäjän identiteetillä. Nyt käyttäjät joutuivat luomaan ensin identiteetin

LIS -palveluun ja vielä erillisen identiteetin Floating Note -palveluun, jonka profiiliin LIS -käyttäjätunnus tallennettiin.

LIS -palveluun rekisteröityneet käyttäjät koostuvat paikkatiedon käyttöään hallitsevista käyttäjistä. Paikkatietoiset sovellukset voisivat tunnistaa käyttäjiä myös LIS -palvelun käyttäjätunnuksien perusteella. Käyttätunnus voitaisiin varmistaa vaikkapa LIS -palveluun liitetyn RADIUS -autentikointipalvelun avulla.

Myös MUPE -sovelluksissa jouduttiin miettimään miten käyttäjä ja käyttäjän päätelaite yhdistetään toisiinsa. Mikäli paikan määrityksessä käytetään LIS -palvelun käyttäjätunnuksia, voitaisiin samaa käyttäjätunnusta käyttää myös MUPE -sovelluksissa suoraan. Nyt kuitenkin toteutuksissa päädyttiin erillisiin käyttäjätileihin sekä MUPE että Floating Note -sovelluksissa. Molempien käyttäjätiliin määriteltiin joko käyttäjän LIS -tunnus tai päätelaitteen MAC -osoite. Käyttäjähallinta tulisi ottaa huomioon aina käsiteltäessä henkilökohtaisia tietoja.

4.2 Paikkatietohavaintojen käyttö

LIS -paikkatietopalvelun ongelmaksi havaittiin paikkatietomallin puute. Soluhavaintojen kytkeminen varsinaiseen sijaintiin tulisi tehdä käyttämällä paikkatietomalleja, joita esiteltiin kappaleessa 2.1. Toteutetussa palvelussa paikkatietomäärittelyjä lisättiin kuitenkin vasta jälkeenpäin (esimerkiksi koordinaattien yhdistäminen solusijainteihin). Rakennetuissa esimerkkisovelluksissa oli siten vaikea yhdistää soluhavaintoja varsinaiseen paikkatietoon. Puutteeksi koettiin myös soluhavaintojen erottelu. Esimerkiksi tilanteessa, jossa kaksi WLAN -tukiasemaa ovat lähekkäin samalla alueella (esimerkiksi silloin kun hetkellisesti halutaan kasvattaa verkon kapasiteettia, vaikkapa tapahtuman yhteydessä jolloin WLAN -verkolla on paljon käyttäjiä), voidaan käyttäjän paikasta saada kaksi soluhavaintoa, jotka pitäisi yhdistää samaan paikkaan. Nyt kuitenkin nämä soluhavainnot välitetään suoraan sovellukselle, joka voi tulkita havaintoja erillisinä sijainteina. Esimerkiksi Floating Note -järjestelmässä nämä havainnot loisivat kaksi erillistä, samalla alueella olevaa, keskustelualuetta, mikä ei ole toivottua. Paikkatiedon käyttö sovelluksissa tulisikin mieluiten perustua paikkatietomalliin eikä pelkästään tiettyyn sovelluskohtaiseen lähteeseen, koska sovelluskohtaisen havainnointitieto saattaa tuoda esiin odottamattomia ongelmia. Paikkatietomalli taas määrittää kehyksen, jossa sovellus voi toimia.

LIS -järjestelmän testiluontoisuuden vuoksi käytettiin pelkästään LIS:n pyyntömetodeja paikkatiedon välittäjäkomponenteissa MUPE -ympäristön yhteydessä. Ilmoitusmetodit ol-

tiin alunperin luotu seurantasovellusten käyttöön, mutta näitä sovelluksia ei oltu kehitetty WLPR.NET -projektin puitteissa, joten ilmoitusmetodien kehitystä ei jatkettu. Ilmoitusmetodien nykyisessä toteutuksessa PostgreSQL -tietokannasta käsin laukaistava SOAP -ilmoituksen lähetys hidastaa myös itse tietokannan toimintaa. Liitteessä 2 esitetty välittäjäskripti laukaisee LIS -palvelussa HTTP -pyynnön ilmoituksen lopulliselle kohteelle. Vasta saataessa HTTP -pyyntöön vastaus jatkuu LIS -palvelun, sekä myös välittäjäskriptin suoritus. Tämä aiheuttaa hidastumista myös tietokannan toiminnalle. Toimintaa olisikin pitänyt parantaa jonotusjärjestelmällä, jossa LIS -palvelu vain vastaanottaisi ilmoituspyynnön ja sallisi PostgreSQL -tietokannassa olevan välittäjäskriptin nopean paluun. LIS -palvelussa taas huolehdittaisiin esimerkiksi tilanteista, jossa ilmoituksen kohde ei olekaan saatavissa.

Pelkkien pyyntömetodien käyttö aiheutti enemmän liikennettä LIS -palvelimelle kuin olisi ollut tarpeellista, sillä paikkahavaintoja pyydettiin lyhyin (esimerkiksi viisi sekuntia) väliajoin. Toisaalta kuormitus painottui enemmänkin itse tilannetiedon välittäjäkomponenttiin, joka toimi palvelimena MUPE -sovelluksille. Näin tapahtui etenkin opetuskäytössä CodeCamp -tapahtumissa, jossa yhtäaikaisesti välityskomponenttia saattoi käyttää 5-10 MUPE sovellusta.

4.3 MUPE:n rajoitukset

Vaikka MUPE ympäristö on tarkoitettu sovelluskehitykseen mobiileilla päätelaitteilla, on ympäristön toteutuksessa kuitenkin seikkoja, jotka heikentävät MUPE:n käyttöä mobiilissa ympäristössä. Käyttöliittymien rakentamiseen tarkoitettu XML -skriptikieli johtaa monimutkaisemmilla käyttöliittymillä suurehkojen XML -dokumenttien muodostamiseen. Suuret XML -dokumentit ovat hitaita siirtää esimerkiksi GPRS (General Packet Radio Service) -yhteyden yli. XML -dokumenttien kokoa on kuitenkin pyritty pakkauksella pienentämään myöhemmissä MUPE -alustan versioissa.

Lisäksi skriptikieli on osittain suppea toiminnallisuuksiltaan. Esimerkiksi aritmeettisissa lausekkeissa ei ole automaattisia muuttujia vaan välitulokset joudutaan tallentamaan erillisiin XML -elementtien attribuutteihin. Tämä hidastaa sovelluskehitystä. Skriptikielen opettelu vie myös aikaa, sillä XML -pohjaisen scriptikielen toimintalogiikka poikkeaa perinteisistä funktionaalisista ohjelmointikielistä. Tällä on kuitenkin hyvätkin puolensa. Esimerkiksi toiminnallisuuksien perintä elementtien kesken on helposti toteutettavissa. Eräs kehitystä hidastava tekijä oli myös skriptikielen dokumentaatio, joka tämän diplomityöprojektin aikana oli vaihtelevaa. Kaikkia toiminnallisuuksia ei oltu dokumentoitu. Lisäksi

alusta oli jatkuvasti kehityksen alla, joten erilaisia virheitä huomattiin runsaasti. Kuitenkin on todettava, että alustalla tehtyjen ohjelmistojen vaativuus alustaa kohtaa ei ollut aluksi tiedossa. Alustan rajat tulivat siis vastaan esimerkiksi nopeatempoisten sovellusten kehityksessä

MUPE -alustan jakelu tehtiin aluksi sekä asiakas- että palvelinohjelmiston yksittäisinä pakettijulkaisuina. Myöhemmin palvelin jaettiin kahteen kirjastoon, core ja content -osuuksiin. Core -kirjasto sisälsi MUPE:n väliohjelmisto -toiminnallisuuden, eli kommunikointimetodit palvelimen eri osien välillä. Content -osa sisälsi alkuperäisen jakelun toteuttaman yksinkertaisen MUD (Multi User Dungeon) -tyyppisen pelimaailman. Asiakasohjelma jaettiin MIDP -alustan mukaisiin versioihin, MIDP1 ja MIDP2 -paketeiksi. MUD -pelimaailman sisällyttämisen tarkoitus oli nopeuttaa palvelujen kehitystä, sillä se sisälsi joitakin yleiskäyttöisiä toiminnallisuuksia, kuten pelaajahahmon ja pelimaailman huoneiden luonnin. Tämä saattoi antaa käsityksen, että MUPE -alusta oli sovelias vain MUD -pelien rakentamiseen, vaikka näin ei ollut. Kuitenkin MUPE -jakelusta puuttui yleisiä, monen pelaajan peleissä tärkeitä käyttäjien hallinnointityökaluja. Tälläisiä olisi ollut pelaajien listaaminen ja poistaminen ylläpidon toimesta.

4.4 Code Camp -tapahtumat

Code Camp -tapahtumat ovat LTY:n tietoliikenteen laitoksen käyttämä opetusmetodi, jossa pyritään intensiivisellä ja tehokkaalla ajankäytöllä opettamaan ja perehdyttämään opiskelijat kulloinkin valittuun ohjelmistokehitysalustaaan. MUPE -alustaa käytettiin hyväksi kahdessa Code Camp -tyyppisessä tapahtumassa. Näistä ensimmäinen järjestettiin elokuussa 2004, WAWC04 (2nd Workshop on Applications of Wireless Communications) -konferenssin yhteydessä. Tällöin käytetiin hyväksi sekä Ekahau- että LIS -järjestelmistä paikkatietoa tuottavia tilannetiedon tuottajakomponentteja. WAWC04 -tapahtumassa toteutettiin 24 tunnin aikana paikkatietoisia sovelluksia. Tämän tapahtuman järjestelyjä, kehitettyjä sovelluksia ja niiden arviointia on käyty läpi tarkemmin julkaisussa Rapid Prototyping of Location-Based Games with the Multi-User publishing Environment Application Platform [33]. Tilannetiedon tuottajia käytettiin myös tammikuussa 2005 järjestetyllä sovelluskehityksen erikoiskurssilla. Kurssi pidettiin viikon intensiivikurssina, jonka aikana opiskelijat tutustuivat pelisuunnitteluun sekä tilannetiedon käyttöön pelisovelluksissa. Kurssia varten kehitettiin tilannetiedon tuottajat Tunturin recumbent -kuntopyörälle sekä RFID -lukijalle että ilmatieteen laitoksen sääpalvelulle.

Tapahtumien avulla, etenkin WAWC'04:n yhteydessä, pyrittiin testaamaan MUPE -alustaa sovelluskehitysympäristönä, joka mahdollistaisi tilannetietoisten sovellusten nopean luomisen. Tapahtuman aikana noin 20 hengen osanottajajoukko, jaettuna viiteen ryhmään, loi viisi sovellusta, jotka käyttivät erityisesti Ekahau -järjestelmän tuottamaa paikkatietoa. Tapahtuman aikana toimin ylläpitäjänä tilannetiedon välittäjäkomponenteille sekä avustin ja neuvoin ryhmiä MUPE -sovelluskehityksessä. Tapahtuman jälkeen ryhmiä haastateltiin vapaaluontoisesti MUPE -alustan käytöstä.

Ryhmien mielestä paikkatiedon käyttö MUPE -alustalla oli helppoa, sillä paikkatieto oli pitkälle jalostettu MUPE -alustan avulla (tieto oli käsiteltävissä Java -objektina). Vaikkakin toimivia sovelluksia pystyttiin kehittämään 24 tunnin aikana, suurimmaksi ongelmaksi havaittiin uuden alustan opettelu. Etenkin käyttöliittymien luonti XML -skripteillä oli aluksi hidasta. XML -skripteillä luotujen käyttöliittymien toiminnallisuuksien testaukseen ei ollut vielä erillisiä työkaluja. Tämä taas hidasti virheiden havainnointia ja siten kehitystä. MUPE -alustan sovellusmahdollisuudet olivat myös vielä osittain rajoittuneita. Esimerkiksi käyttäjän ruudun tilanteen päivitys vaati jatkuvaa muutosten kyselyä MUPE -palvelimelta, jolloin nopeita yksittäisiä päivityksiä pystyttiin tekemään vain ruudun päivityskyselyiden lähetysaikavälien tarkkuudella. Tämä johtui asiaskasohjelmistojen yhteyksiin käytetystä tilattomasta HTTP -protokollasta, jolloin asiakasohjelmistolle ei voitu tehdä suoria päivityksiä. Ongelma kuitenkin korjaatui myöhemmissä MUPE -alustan versioissa HTTP -yhteyksien rinnalle rakennetulla yhteyden tilan säilyttävällä TCP/IP -protokollalla. Kyseinen parannus on saatavilla asiaskasohjelmiston MIDP 2 -versiossa.

Luku 5

Johtopäätökset

Tässä diplomityössä käsiteltiin paikka- ja tilannetiedon hyödyntämistä sovelluksissa. Työssä perehdyttiin sekä paikkatiedon määrittelyyn, paikannusjärjestelmiin ja paikkatiedon hyödyntämistä tukeviin ohjelmistorajapintoihin. Työssä perehdyttiin myös tilannetiedon määrittelyyn sekä tilannetietoa käsitteleviin järjestelmiin. Työssä oli tavoitteena rakentaa erilaisia paikka- ja tilannetietoisten sovellusten kehitystä tukevia järjestelmiä, sekä tuottaa näiden avulla sovelluksia.

Työn tuloksena toteutettiin WLPR.NET -projektin parissa solupaikannuspalveluun palvelurajapinta (Location Information Service, LIS). Palvelurajapinnasta esitettiin julkaisu ([25]) Euromedia'2005 konferenssissa. Palvelurajapintaa hyödyntämään toteutettiin erikoistyönä paikkatietoinen viestintäsovellus, josta esitettiin myös julkaisu ([32]) WAWC04 konferensissa.

WLPR.NET -verkkoa hyödynnettiin myös Ekahau EPE (Ekahau Positioning Engine) -sovelluksen avulla paikkatiedon tuottamisessa. Sovelluskehitysalustana toimi MUPE (Multi User Publishing Environment) -alusta, johon toteutettiin sekä LIS, että Ekahau EPE -järjestelmistä paikkatietoa tuottavat komponentit. Näitä komponenttejä käytettiin WAWC04 konferenssin yhteydessä järjestetyssä CodeCamp -tapahtumassa. WAWC04:n CodeCamp -tapahtuman tuloksista esitettiin julkaisu ([33]) IE 05 (Intelligent Environments 2005) konferensissa. MUPE -alustalle toteutettiin myös laajemmin määriteltyä tilannetietoa (säätieto, kuntopyörästä saatu harjoitustieto sekä RFID -tunnistetieto) tuottavia komponentteja. Näitä tilannetiedon tuottajakomponentteja käytettiin MUPE -sovelluskehityksessä AMPERS -projektin parissa ja LTY:n tietoliikennetekniikan laitoksen Sovelluskehityksen erikoistyökurssilla opetuksessa. AMPERS -projektin parissa toteutettiin myös edellämainit-

tuja tilannetiedon tuottajakomponentteja (sää- ja LIS -solupaikkatieto) hyödyntävä tilannetietoinen pelisovellus.

Paikkatietoisten sovellusten kehitykseen tehdään jatkuvasti standardointityötä sekä erilaisia työkaluja. Onkin nähtävissä, että paikkatietoisten sovellusten kehitys hyötyy tästä työstä. Paikkatiedon saatavuus kuitenkin vaihtelee paikannusjärjestelmän käytön mukaan. Paikkatiedon tuottamiseen, hallintaan ja käsittelyyn tarvitaan monia eri järjestelmiä, jotka ottavat myös yksityisyyden suojan huomioon. Paikkatietoisten palvelujen kehittäminen esimerkiksi WLAN -verkkoihin voidaan tämän työn perusteella todeta olevan suuritöinen projekti, etenkin jos hallittavana on kokonainen sovellusketju paikkatiedon keruusta, hallinnoinnista, välittämisestä ja hyödyntämisestä sovelluksissa. Paikkatietorajapinnoissa todettiin kuitenkin WSDL -kuvausten ja SOAP -protokollan käytön olevan toimiva ratkaisu joka mahdollistaa sovellusten kehitysten usealla eri alustalla. Valmiit tuotteet, kuten Ekahau, tarjoavat kuitekin paljon jalostetumpaa ja tarkempaa paikkatietoa.

Tilannetietoisten sovellusten kehittäminen ei ole vielä laajalti yleistynyttä. Tällaiset sovellukset käsittelevät usein monista eri tietolähteistä kerättyä tietoa, joka liittyy sovelluksen käyttötilanteeseen. Tilannetietoisten sovellusten kehittäminen olisi myös suuritöistä ilman kehitystä tukevia ympäristöjä, kuten MUPE. Tilannetiedon välittäminen MUPE -ympäristöön on alustan ulkopuolinen toiminto, joka tapahtuu muuntamalla tieto CEP -protokollan muotoon. MUPE -alustalle tilannetiedon tuomisessa ovatkin tilannetiedon tuottajakomponentit tärkeässä asemassa.

Tilannetiedon simulointi muodostuu tärkeäksi asiaksi tilannetietoisten sovellusten kehityksessä. Tilannetietoisia sovelluksia pitäisi pyrkiä testaamaan lopullisessa ympäristössä, jotta voitaisiin tehdä johtopäätöksiä sovelluksen toimivuudesta aidoilla päätelaitteilla ja aidoissa käyttötilanteissa. Testiympäristön järjestely on kuitenkin vaikeaa etenkin, jos sovelluksessa käytetään monia tiettyyn paikkaan sitovia tilannetietoisia elementtejä. Esimerkiksi paikkatiedon saatavuutta rajoittaa käytetyn paikannusjärjestelmän kattavuus. Tällöin alustat kuten MUPE tulevat tärkeiksi sovelluskehityksen kannalta.

Ilman tilannetietoista sovelluskehitystä tukevia järjestelmiä olisi tämäntyyppisten sovellusten kehittäminen huomattavasti vaikeampaa. Monen eri tilannetiedon lähteen (esimerkiksi paikka, säätila, mahdolliset tunnistejärjestelmät kuten RFID) yhtäaikainen hallinta yhdessä sovelluksessa kasvattaisi sovelluksen kokoa huomattavasti. Välittäjäalustoilla kuten MUPE saadaan ohjelmistokehitystä keskitettyä itse sovelluksen päätoimintoihin. Kuitenkin on todettava, että sovelluskehitystä helpotettaessa myös helposti rajataan toimintaympäristöä liian suppeaksi. Parhaimmillaan erilaiset tilannetietoista sovelluskehitystä avusta-

vat työkalut ovat itsenäisiä komponentteja joita voitaisiin helposti lisätä mahdollisesti jo olemassaoleviin sovelluksiin.

Kirjallisuutta

[1] Roy Want, Andy Hopper, Veronica Falcao, and Jon Gibbons. The active badge location system. Technical Report 92.1, ORL, 24a Trumpington Street, Cambridge CB2 1QA, 1992. URL `http://citeseer.ist.psu.edu/want92active.html(tarkistettu5.10.2005)`.

[2] Liikenne ja viestintäministeriö. *Sähköisen viestinnän tietosuojalaki*. WWW (tarkistettu 18.8.2005), `http://www.finlex.fi/fi/laki/ajantasa/2004/20040516`, syyskuu 2004.

[3] Tekniikan Sanastokeskus ry. *Paikannussanasto*. Tekniikan Sanastokeskus ry, `http://www.tsk.fi/fi/info/paikannussanasto.pdf`, 2002. ISBN 952-9794-16-9. ISSN 0359-5390.

[4] Maanmittauslaitos. *Paikkatietotekniikan perusteet - KKJ*. WWW (tarkistettu 23.8.2005), `http://www.nls.fi/ptk/perusteet/paikkatieto/sijainti/kkj.html`, 1996.

[5] Peter H. Dana. *Geodetic Datum Overview*. WWW (tarkistettu 29.9.2005), `http://www.colorado.edu/geography/gcraft/notes/datum/datum_f.html`, 2003.

[6] EUROCONTROL and IfEN. *WGS 84 Implementation Manual*. WWW (tarkistettu 1.7.2005), `http://www.wgs84.com/files/wgsman24.pdf`, 1998.

[7] Antti Rainio. Paikannus mobiilipalveluissa ja sovelluksissa. Technical report, TEKES, 2003. ISBN 952-457-131-5.

[8] Akio Yasuda Falin Wu, Nobuaki Kubo. Performance evaluation of gps augmentation using quasi-zenith satellite system. IEEE Transactions on Aerospace and Electronic Systems, lokakuu 2004.

[9] Ekahau. *Ekahau*. WWW (tarkistettu 23.8.2005), `http://www.ekahau.com/`, 2005.

[10] Bluetooth Local Positioning Working Group. *Local Positioning Profile.* heinäkuu 2003. Saatavilla rekisteröitymällä osoitteessa www.bluetooth.org.

[11] Josef Hallberg and Marcus Nilsson. Positioning with bluetooth, irda and rfid. Master's thesis, Luleå University of Technology, `http://epubl.luth.se/1402-1617/2002/125/LTU-EX-02125-SE.pdf` (tarkistettu 5.10.2005), 2002.

[12] Raine Koponen. Ennakoivan bluetooth-verkon hyödyntäminen paikkatietoon pohjautuvan opastusjärjestelmän toteutuksessa. Master's thesis, Lappeenrannan Teknillinen Yliopisto, 2004.

[13] Urs Hengartner and Peter Steenkiste. Implementing access control to people location information. In *SACMAT '04: Proceedings of the ninth ACM symposium on Access control models and technologies*, pages 11–20. ACM Press, New York, NY, USA, 2004. ISBN 1-58113-872-5.

[14] J. Hightower, B. Brumitt, and G. Borriello. The location stack: A layered model for location in ubiquitous computing, 2002. URL `citeseer.ist.psu.edu/hightower02location.html`.

[15] Open Geospatial Consortium. *Web Coordinate Transformation Service.* WWW (tarkistettu 25.8.2005), `https://portal.opengeospatial.org/files/?artifact_id=8847`, tammikuu 2005.

[16] Refractions Research. *PostGIS.* WWW (tarkistettu 30.6.2005), `http://postgis.refractions.net`, 2005.

[17] PostgreSQL Global Development Group. *PostgreSQL: The world's most advanced open source database.* WWW (tarkistettu 30.6.2005), `http://www.postgresql.org`, 2005.

[18] Anind K. Dey. *Providing Architectural Support for Building Context-Aware Applications.* WWW (tarkistettu 25.8.2005), `http://www.cc.gatech.edu/fce/ctk/pubs/dey-thesis.pdf`, marraskuu 2000.

[19] Harri Lakkala. *Context Exchange Protocol Specification.* WWW (tarkistettu 1.7.2005), `http://www.mupe.net/inet/mupe/AKPMedia.nsf//Resources/context_exchange_protocol_1_0.pdf/\protect\T1\textdollarfile/context_exchange_protocol_1_0.pdf`, 2003.

[20] Ari Koivisto. *MUPE Server Technical Document.* WWW (tarkistettu 12.8.2005),

http://www.mupe.net/inet/mupe/AKPMedia.nsf/Resources/MUPE_server_tech.pdf/\protect\T1\textdollarfile/MUPE_server_tech.pdf, huhtikuu 2004.

[21] Ari Koivisto Riku Suomela, Eero Räsänen. *MUPE Context Programming Manual.* WWW (tarkistettu 15.8.2005), http://www.mupe.net/inet/mupe/AKPMedia.nsf/Resources/MUPE_Context_Programming_Manual.pdf/\protect\T1\textdollarfile/MUPE_Context_Programming_Manual.pdf, huhtikuu 2004.

[22] Matti Juutilainen, Jouni Ikonen, and Jari Porras. Evaluation of a next generation public wireless multi-isp network. In *Proceedings of* 27^{th} *conference on Local Computer Networks*, pages 405–414. IEEE, Tampa, Florida, 6-8 November 2002.

[23] Antti Seppänen. Paikkatiedon kerääminen ja hyödyntäminen wlan-verkossa. Master's thesis, Lappeenrannan Teknillinen Yliopisto, 2002.

[24] Mikko Hietala. Paikkatiedon hyödyntäminen jabber-pikaviestintäohjelmistossa. Master's thesis, Lappeenrannan Teknillinen Yliopisto, 2003.

[25] Kimmo Koskinen and Jouni Ikonen Radek Spáčil. Location information system – a description of a positioning middleware system. In *Euromedia'2005.* ISBN 90-77381-17-1.

[26] Pavel Kulchenko. *SOAP::Lite Home.* WWW (tarkistettu 11.7.2005), http://soaplite.com/, 2005.

[27] W3C: Web Services Architecture Working Group. *Web Services Architecture.* WWW (tarkistettu 4.10.2005), http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/, 2004.

[28] 802.11 IEEE Wireless LAN Working Group. Part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. Technical report, ANSI/IEEE, 1999.

[29] Tim Bunce and Ilya Sterin. *Perl DBI - dbi.perl.org.* WWW (tarkistettu 11.7.2005), http://dbi.perl.org/, 2005.

[30] *Practical PostgreSQL.* O'Reilly, http://www.commandprompt.com/ppbook/ (tarkistettu 5.10.2005), 2002.

[31] The PostgreSQL Global Development Group. *PostgreSQL 7.2.8 Documentation.* WWW (tarkistettu 12.7.2005), http://www.postgresql.org/docs/7.2/interactive/index.html, 2005.

[32] Markku Multaharju, Kimmo Koskinen, and Jouni Ikonen. Floating note—a location based messaging application. In *Proceedings of 2nd Workshop on Applications of Wireless Communications.*

[33] Kari Heikkinen Riku Suomela, Kimmo Koskinen. Rapid prototyping of location based games with the multi-user publishing environment application platform. In *Intelligent Environments 2005.* ISBN 0-86341-519-9.

Liite 1

PL/pgSQL trigger -funktio

```
--
-- LUT, Kimmo Koskinen
-- kimmo.m.koskinen@lut.fi
--
-- Location Information System (LIS)
--
-- Function
-- loc_tgfunc(): Trigger condition function for LIS triggers
--
-- This function is a PostgreSQL trigger function. It checks if any update
-- has created a situation where for any user, the new and old location
-- differ. If this is true, the table loc_trigger is gone through for any
-- outstanding triggers for the particular user. If there is a trigger for
-- the user, the DB calls lis which in turn fires the monitoring
-- trigger from the service.
--
-- Parameters: Usual parameters which are given for trigger functions

DECLARE
-- declare a row variable from the loc_trigger table
row_data loc_trigger%ROWTYPE;
-- variable for the location description
l_id text;
-- variable for the users nickname
nickname text;
-- variable for the trigger id
```

```
tg_id integer;
-- variable for the service id
s_id text;
-- variables for the service uri and proxy
uri text;
proxy text;
BEGIN
-- if there has been a change in users location
IF NEW.ap != OLD.ap
THEN
-- loop through all the triggers who are following the user who's
-- location has changed
FOR row_data IN SELECT * FROM loc_trigger WHERE loc_trigger.mac = OLD.mac LOOP
-- fetch the description of the location
SELECT INTO l_id loc_id FROM loc_desc WHERE ap = NEW.ap;
-- fetch the nickname of the user
SELECT INTO nickname nick FROM user_data WHERE mac = OLD.mac;
-- fetch the trigger id for this trigger
tg_id := row_data.trig_id;
-- fetch the service id who placed this trigger
s_id := row_data.serviceid;
-- fetch the uri for the handler
uri := row_data.uri;
-- fetch the proxy for the handler
proxy := row_data.proxy;
-- SELECT INTO proxy proxy FROM loc_trigger WHERE mac = macaddr(NEW.mac);
-- call the LIS trigger handler
PERFORM lis_trigger_perl(nickname, l_id, tg_id, s_id, uri, proxy);
END LOOP;
 END IF;
-- allow the modification to be made
RETURN NEW;
END;
```

Liite 2

PL/Perl välitysfunktio

```
#
# LUT, Kimmo Koskinen
# kimmo.m.koskinen@lut.fi
#
# Location Information System (LIS)
#
# Function
# lis_trigger_perl(): Calls HandleTrigger() from the LIS which fires the
#                     trigger callback (FireTrigger()) from the service
#
# Parameters
# $_[0]: nick               # Nickname of the user which fired the trigger
# $_[1]: description        # The new location_id for the user
# $_[2]: trigger id         # ID number of the trigger which was fired
# $_[3]: service id         # Service which placed the trigger
# $_[4]: URI                # URI from where LIS calls FireTrigger()
# $_[5]: proxy              # Proxy through which LIS calls FireTrigger()
#

#import SOAP library
use SOAP::Lite;

#LIS error code definitions
$CERROR => 0;
$COK => 1;
```

```
#parameters for the sript are:
$nick = $_[0];        # nickname of the user
$location_id = $_[1]; # location id
$trigger_id = $_[2];  # trigger id
$service_id = $_[3];  # service id
$uri = $_[4];         # uri
$proxy = $_[5];       # proxy

my $res;

#establish a connection to the SOAP server
my $lite = SOAP::Lite
    -> uri('urn:LIS')
    -> proxy('http://localhost:1234/')
    ;

#call the trigger handler from LIS
my $som = $lite->HandleTrigger(SOAP::Data->name(nick => $nick),
       SOAP::Data->name(location_id => $location_id),
       SOAP::Data->name(trigger_id => $trigger_id),
       SOAP::Data->name(service_id => $service_id),
       SOAP::Data->name(uri => $uri),
       SOAP::Data->name(proxy => $proxy));

if ($som->fault) {
$res = $CERROR;
} else {
  $res = $som->valueof('//Envelope/Body/HandleTriggerResponse/success_code')
}

$lite->close;
return "$res";
```

Liite 3

MUPE Context Script säätiedolle

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!--
   Muurityypit tunnistetaan numerolla, joka välitetään
   säätiedon vaikutuksen laukaisevalle clientDoDamage
   - metodille. Numeroiden merkitykset ovat seuraavat:

   Olki: 128
   Jää:  129
   Puu:  130
   Kivi: 131
-->
<script id='fmi.fi'
  xmlns='http://www.nokia.com/ns/cep/script/1.0/'
  xmlns:cep='http://www.nokia.com/ns/cep/1.0/'>
  <if>
    <inRange>
      <atomRef name='Weather' ref='temperature' userId='*'>
      </atomRef>
      <double>10</double>
      <double>20</double>
    </inRange>
    <actions>
      <mupeCall>
<![CDATA[2::clientDoDamage {${userId}} {129} {-2}]]>
      </mupeCall>
    </actions>
```

```
  </if>
  <elseIf>
    <greater>
      <atomRef userId='*' name='Weather' ref='temperature'>
      </atomRef>
      <double>30</double>
    </greater>
    <actions>
      <mupeCall>
<![CDATA[2::clientDoDamage {${userId}} {129} {-4}]]>
      </mupeCall>
    </actions>
  </elseIf>
  <elseIf>
    <inRange>
      <atomRef userId='*' name='Weather' ref='temperature'>
      </atomRef>
      <double>20</double>
      <double>30</double>
    </inRange>
    <actions>
      <mupeCall>
<![CDATA[2::clientDoDamage {${userId}} {129} {-3}]]>
      </mupeCall>
    </actions>
  </elseIf>
  <elseIf>
    <inRange>
      <atomRef userId='*' name='Weather' ref='temperature'>
      </atomRef>
      <double>0</double>
      <double>10</double>
    </inRange>
    <actions>
      <mupeCall>
<![CDATA[2::clientDoDamage {${userId}} {129} {-1}]]>
      </mupeCall>
    </actions>
  </elseIf>
  <elseIf>
    <inRange>
```

```
      <atomRef userId='*' name='Weather' ref='temperature'>
      </atomRef>
      <double>-10</double>
      <double>-20</double>
    </inRange>
    <actions>
      <mupeCall>
<![CDATA[2::clientDoDamage {${userId}} {129} {1}]]>
      </mupeCall>
    </actions>
  </elseIf>
  <elseIf>
    <inRange>
      <atomRef userId='*' name='Weather' ref='temperature'>
      </atomRef>
      <double>-20</double>
      <double>-30</double>
    </inRange>
    <actions>
      <mupeCall>
<![CDATA[2::clientDoDamage {${userId}} {129} {2}]]>
      </mupeCall>
    </actions>
  </elseIf>
  <elseIf>
    <greater>
      <atomRef userId='*' name='Weather' ref='temperature'>
      </atomRef>
      <double>30</double>
    </greater>
    <actions>
      <mupeCall>
<![CDATA[2::clientDoDamage {${userId}} {129} {3}]]>
      </mupeCall>
    </actions>
  </elseIf>
  <elseIf>
    <inRange>
      <atomRef userId='*' name='Weather' ref='humidity'>
      </atomRef>
      <int>60</int>
```

```
      <int>80</int>
    </inRange>
    <actions>
      <mupeCall>
<![CDATA[2::clientDoDamage {${userId}} {128} {-1}]]>
      </mupeCall>
    </actions>
  </elseIf>
  <elseIf>
    <inRange>
      <atomRef userId='*' name='Weather' ref='humidity'>
      </atomRef>
      <int>80</int>
      <int>100</int>
    </inRange>
    <actions>
      <mupeCall>
<![CDATA[2::clientDoDamage {${userId}} {128} {-2}]]>
      </mupeCall>
      <mupeCall>
<![CDATA[2::clientDoDamage {${userId}} {130} {-1}]]>
      </mupeCall>
    </actions>
  </elseIf>
  <elseIf>
    <inRange>
      <atomRef userId='*' name='Weather' ref='windspeed'>
      </atomRef>
      <double>2</double>
      <double>4</double>
    </inRange>
    <actions>
      <mupeCall>
<![CDATA[2::clientDoDamage {${userId}} {128} {-1}]]>
      </mupeCall>
    </actions>
  </elseIf>
  <elseIf>
    <inRange>
      <atomRef userId='*' name='Weather' ref='windspeed'>
      </atomRef>
```

```
      <double>4</double>
      <double>8</double>
    </inRange>
    <actions>
      <mupeCall>
<![CDATA[2::clientDoDamage {${userId}} {128} {-2}]]>
      </mupeCall>
    </actions>
  </elseIf>
  <else>
    <greater>
      <atomRef userId='*' name='Weather' ref='windspeed'>
      </atomRef>
      <double>8</double>
    </greater>
    <actions>
      <mupeCall>
<![CDATA[2::clientDoDamage {${userId}} {128} {-3}]]>
      </mupeCall>
      <mupeCall>
<![CDATA[2::clientDoDamage {${userId}} {130} {-1}]]>
      </mupeCall>
      <mupeCall>
<![CDATA[2::clientDoDamage {${userId}} {129} {-1}]]>
      </mupeCall>
    </actions>
  </else>
</script>
```

Liite 4

MUPE Context Script LIS -järjestelmän solupaikkatiedolle

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<script id='lis'
  xmlns='http://www.nokia.com/ns/cep/script/1.0/'
  xmlns:cep='http://www.nokia.com/ns/cep/1.0/'>
  <if>
    <atomChanged>
      <atomRef userId='*' name='LISLocation'>
      </atomRef>
    </atomChanged>
    <actions>
      <mupeCall>
<![CDATA[2::clientSetLocation {${userId}} {${LISLocation::location}}]]>
      </mupeCall>
    </actions>
  </if>
</script>
```