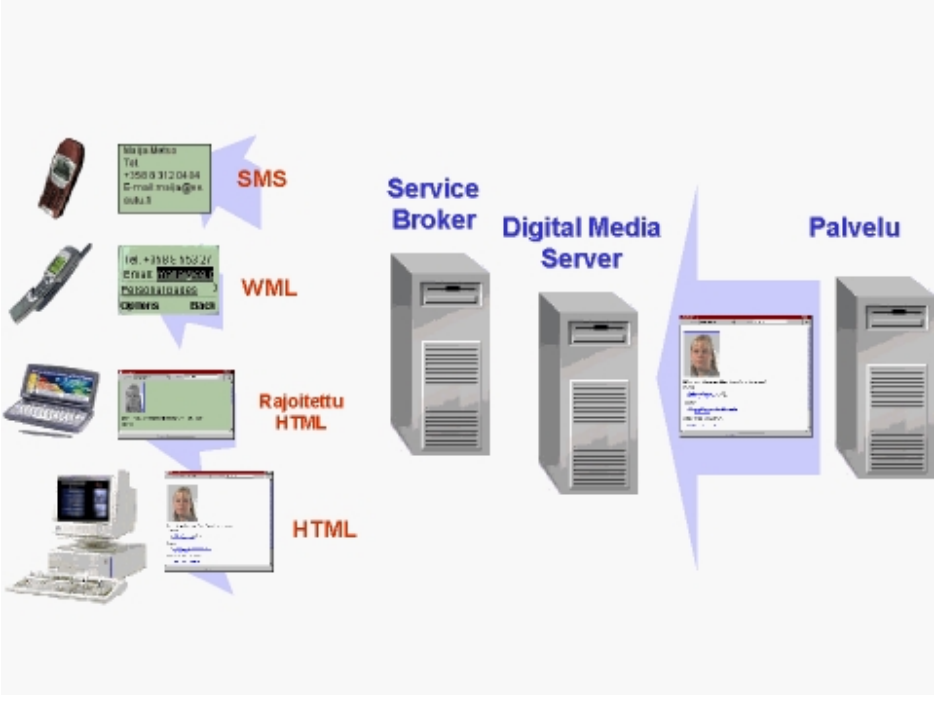
2088

VTT TIEDOTTEITA

VTT TIEDOTTEITA

Jari Korva

Adaptiivisten verkkopalvelujen käyttöliittymät



VALTION TEKNILLINEN TUTKIMUSKESKUS ESPOO 2001

VTT TIEDOTTEITA – MEDDELANDEN – RESEARCH NOTES 2088

Adaptiivisten verkkopalvelujen käyttöliittymät

Jari Korva
VTT Elektroniikka

VALTION TEKNILLINEN TUTKIMUSKESKUS
ESPOO 2001

ISBN 951–38–5810–3 (URL: http://www.inf.vtt.fi/pdf/)
ISSN 1235–0605 (URL: http://www.inf.vtt.fi/pdf/)

JULKAISIJA – UTGIVARE – PUBLISHER

Valtion teknillinen tutkimuskeskus (VTT), Vuorimiehentie 5, PL 2000, 02044 VTT
puh. vaihde (09) 4561, faksi (09) 456 4374

Statens tekniska forskningscentral (VTT), Bergsmansvägen 5, PB 2000, 02044 VTT
tel. växel (09) 4561, fax (09) 456 4374

Technical Research Centre of Finland (VTT), Vuorimiehentie 5, P.O.Box 2000, FIN–02044 VTT, Finland
phone internat. + 358 9 4561, fax + 358 9 456 4374

VTT Elektroniikka, Tietoliikennejärjestelmät, Kaitoväylä 1, PL 1100, 90571 OULU
puh. vaihde (08) 551 2111, faksi (08) 551 2320

VTT Elektronik, Nätteknologier, Kaitoväylä 1, PB 1100, 90571 ULEÅBORG
tel. växel (08) 551 2111, fax (08) 551 2320

VTT Electronics, Telecommunication Systems, Kaitoväylä 1, P.O.Box 1100, FIN–90571 OULU, Finland
phone internat. + 358 8 551 2111, fax + 358 8 551 2320

Toimitus Kerttu Tirronen

VTT Tietopalvelu, Espoo 2001

Korva, Jari. Adaptiivisten verkkopalvelujen käyttöliittymät [User interfaces for adaptive multimedia services]. Espoo 2001, Valtion teknillinen tutkimuskeskus, VTT Tiedotteita – Meddelanden – Research Notes 2088. 71 s. + liitt. 4 s.

Avainsanat adaptive user interfaces, universal access, WWW, abstract user interface model

Tiivistelmä

Tekesin rahoittamassa Princess-projektissa kehitetään palvelualustaa, jolle voidaan rakentaa missä ja milloin tahansa käytettävissä olevia verkkopalveluja – toisin sanottuna järjestelmää, joka mukauttaa palvelun kulloinkin käytettävissä olevaan päätelaitteeseen. Adaptoinnin kohteina ovat sekä media (esimerkiksi kuvien käsittely tiedonsiirron nopeuttamiseksi) että palvelun käyttöliittymä, jota tarkastellaan tässä pro gradu -tutkielmassa.

Työn lähtökohtana on olemassa oleva tieto käyttöliittymistä: niiden käytettävyys, ihmisen ja koneen vuorovaikutustavat sekä syöttö- ja näyttölaitteet. Näihin liittyvät erilaiset päätelaite- ja sovellustyypit, kuten työasemissa ja mukana kuljetettavissa tietokoneissa toimivat työpöytä- ja selainsovellukset sekä matkapuhelimissa toimivat ääniohjatut sovellukset, tekstiviestit ja WAP-selainsovellukset. Lisäksi tutkielmassa kiinnitetään huomiota tietoliikenneverkkojen ja -protokollien vaikutukseen käyttöliittymäratkaisuihin.

Adaptiivisuuden lisäksi merkittävä käsite on esitystapariippumaton käyttöliittymän kuvaustapa eli abstrakti käyttöliittymämalli, toisin sanottuna kuvaus, josta voidaan adaptoida kussakin päätelaitteessa toimiva käyttöliittymä. Abstraktiksi käyttöliittymämalliksi valittiin projektin ensimmäisessä vaiheessa itse kehitetty oliomalli, joka sisälsi yleisimmät käyttöliittymäkomponentit (esimerkiksi valintalista, linkki, nappi). Toteutuksessa esiintyneiden puutteiden takia alettiin myöhemmin etsiä standardinmukaisempaa ratkaisua, joista lupaavimpana valittiin rakenteellisen kuvauskielen, XML:n, sovellus XHTML.

Tutkielman kohteena oleva Princess-järjestelmän osa on toteutettu Java-kielen ja XML-työkalujen avulla. Tuettuja päätelaite- ja käyttöliittymätyyppejä ovat Java-sovellukset (myös puheohjauksella käytettynä), WWW-selaimet, tekstiviestit ja sähköposti sekä jatkossa myös WAP.

Korva, Jari. Adaptiivisten verkkopalvelujen käyttöliittymät [User interfaces for adaptive multimedia services]. Espoo 2001, Technical Research Centre of Finland, VTT Tiedotteita – Meddelanden – Research Notes 2088. 71 p. + app. 4 p.

Keywords adaptive user interfaces, universal access, WWW, abstract user interface model

Abstract

The goal of the Princess project is to research and develop a platform for multimedia services that allows access to the contents regardless of the user's location and the terminal device – i.e. the purpose of the system is to adapt services to match various network and terminal capabilities. This thesis focuses on adapting user interfaces, which constitutes one part of the complete adaptation process.

The starting point of this thesis is the existing knowledge of user interfaces: usability, human-computer interaction types, input/output devices and implementation level technologies for developing user interfaces. Also characteristics of different application types are described, including desktop and browser applications for fixed and mobile terminals as well as short messages and voice-based systems.

In order to achieve adaptivity, a model (i.e. format) that describes the user interface independent of the final presentation modality and platform, is required. This format is called the abstract user interface model or the presentation model – the latter concept is wider because in addition to the user interface, also content needs to be modelled.

An object-based presentation model capable of describing basic UI concepts was developed in the first phase of the Princess-project. Later this model was discarded in favour of a standard-based solution, XHTML, which is a reformulation of HTML in XML.

The Princess system is implemented using Java and XML technologies. Supported terminal/user interface types include WWW and WAP browsers, short messages (SMS) and email. Experiments have been made with graphical Java applications with speech driven input.

Alkusanat

Tulin mukaan Princess-projektiin syksyllä 1998. Siinä vaiheessa järjestelmää oli jo suunniteltu ja osin toteutettukin. Pääsin siis tutustumaan järjestelmään – ja sen puutteisiin – jatkamalla ja ylläpitämällä aiempaa toteutusta sekä osallistumalla järjestelmän osien integrointiin. Tutkielmassa tätä jaksoa kutsutaan ensimmäiseksi vaiheeksi. Toisessa vaiheessa, joka alkoi loppuvuodesta 1998, projektissa alettiin etsiä vaihtoehtoisia ideoita ja toteutustapoja siihen astisten kokemusten pohjalta. Tällöin pääsin tekemään selvitystä XML-teknologioista ja myöhemmin kevätkesästä 1999 lähtien myös toteuttamaan näitä ideoita tutkielmassa kuvaamallani tavalla.

Tämä työ on tehty kokonaan VTT Elektroniikassa ja sen ohjaajina toimivat VTT:sta tekn. lis. Johan Plomp ja tekn. lis. Marko Heikkinen sekä Oulun yliopistosta fil. toht. Kari Kuutti (kesästä 1999 lähtien Teknillinen korkeakoulu). Kiitokset heille lukuisista hyvistä kommenteista. Kiitokset kuuluvat myös kaikille niille noin kymmenelle henkilölle, jotka ovat olleet mukana Princess-projektissa VTT:ssa tai Oulun yliopistossa, erityisesti tekn.yo. Petri Määtälle ja fil. maist. Kim Lempiselle.

Jari Korva
jari.korva@vtt.fi

Sisällysluettelo

Symboliluettelo

API	Application programming interface. Ohjelmointirajapinta.
Appletti	WWW-selaimessa toimiva Java-sovellus.
ATM	Asynchronous transfer mode.
AWT	Abstract window toolkit. Java-käyttöliittymäkirjasto.
B-ISDN	Broadband ISDN. ATM-pohjainen ISDN.
CGI	Common gateway interface. Tapa luoda dynaamisia selainsovelluksia palvelinpäässä.
CLIM	Common Lisp interface manager.
CSS	Cascading style sheets. Dokumentin esitystavan kuvauskieli.
CTI	Computer-telephone integration.
DMS	Digital media server. Princess-järjestelmän komponentti.
DOM	Document object model. Ohjelmointikielineutraali rajapinta XML- ja HTML-dokumenttien käsittelyyn.
DSSSL	Document style semantics and specification language.
DTD	Document type definition. XML- tai SGML-dokumentin oikean rakenteen määrittelytapa.
EDGE	Enhanced data-rates for GSM evolution
GPRS	GSM general packet radio service.
GSM	Global system for mobile communications.
GSM 07.05	AT-komentorajapinta GSM-tekstiviestiominaisuusiin.
HCI	Human-computer interaction.

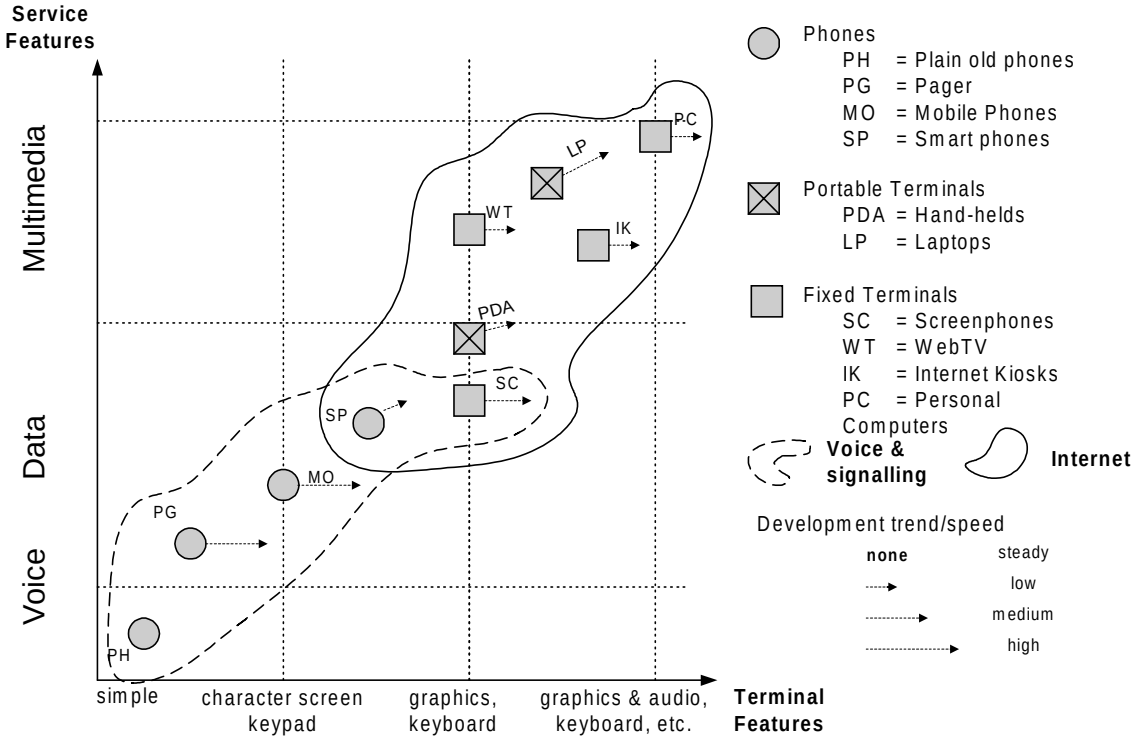
HMD	Head mounted display.
HSCSD	High speed circuit switched data.
HTML	Hypertext markup language.
HTTP	Hypertext transform protocol.
ISDN	Integrated service digital network. Katso B-ISDN and N-ISDN.
IVR	Interactive voice response. Äänisovellustyyppi.
JFC	Java foundation classes. Java-käyttöliittymäkirjasto.
LAN	Local area network.
LISP	List processing. Ohjelmointikieli.
MHEG	Multimedia and hypertext experts group.
N-ISDN	Narrowband ISDN. "Tavallinen" ISDN (64 tai 128kb/s)
Parseri	Jäsennin. Työkalu, joka lukee dokumentin rakenteen ja sisällön.
PDA	Personal digital assistant. Kämmentietokone.
PDF	Adobe portable document format.
RMI	Remote method invocation.
SAX	Simple API for XML. Tapahtumapohjainen parserirajapinta.
ServiceBroker	Princess-komponentti.
Servletti	WWW-palvelimessa toimiva Java-sovellus. Ks. CGI.
SGML	Standardized general markup language.
SLI	Spoken language interface.
SMS	Short message service.

SVG	Scalable vector graphics. XML-pohjainen grafiikkaformaatti.
Swing	Java-käyttöliittymäkirjasto, synonyymi JFC.
Tcl/Tk	Tcl-ohjelmointikieli/Tk-käyttöliittymäkirjasto.
UMTS	Universal mobile telecommunications system.
W3C	World Wide Web Consortium.
WAP	Wireless application protocol.
WML	Wireless markup language.
VoxML	Voice markup language.
WWW	World wide web.
X Window	Graafinen, käyttöjärjestelmäriippumaton ikkunointijärjestelmä.
xDSL	Digital subscriber line.
XHTML	Extensible hypertext markup language.
XML	Extensible markup language.
XML-sovellus	XML-dokumenttityyppi.
XSL	Extensible stylesheet language. XML:n sovellus, joka soveltuu dokumentin ulkoasun esittämiseen.
XSLT	XSL Transformations. Muunnoskieli XML-dokumenteille.

1. Johdanto

Tausta

Tietoliikenteen kehitys on nopeaa – tämä koskee niin teknisiä ominaisuuksia, tarjolla olevia palveluja kuin käyttäjien määrääkin. Kiinteiden verkkojen, kuten Internetin, käyttäjien määrä on lisääntynyt. Tämän lisäksi on tullut ja tulee kokonaan uusia langattomia verkkoja (GSM, 3. sukupolven verkot). Näihin verkkoihin liitetään monenlaisia päätelaitteita yksinkertaisista puhelimista tietokoneisiin. Myös päätelaitteiden ominaisuudet ovat kehittyneet ja erityisesti käyttöön on tullut langattomia laitteita, joissa on kuitenkin monia aiemmin vain työasemista löytyneitä ominaisuuksia (kuva 1).



Kuva 1. Eri päätelaitteet ominaisuuksien mukaan luokitettuna ja arvio niiden ominaisuuksien kehityksestä (CTI 1999).

Erilaisten palvelujen tarjoajat, kuten pankit, kaupat ja informaation tuottajat, ovat huomanneet uusien teknologioiden mahdollisuudet. Olemassa olevia palveluja, siirretään verkkoon siten, että asiakkaat voivat hoitaa asiansa – usein itsepalveluna – paikasta ja ajasta riippumatta puhelimen, Internetin tai tekstiviestien avulla. Lisäksi on alettu ke-

hittää kokonaan uusia palvelumuotoja, jotka perustuvat esimerkiksi langattomuuden ja käyttäjän liikkuvuuden tuomiin mahdollisuuksiin.

Tekninen ja kaupallinen kehitys on luonut tarpeen ratkaisuille, joiden avulla palvelut ovat nykyistä helpommin saatavilla siellä, missä käyttäjä kulloinkin on. Tämä tarkoittaa esimerkiksi sitä, että samoja palveluja voidaan käyttää kotona nopealla verkkoyhteydellä varustetulla työasemalla tai kotimatkalla langattomalla puhelimella. Palvelujen kehittäjän kannalta tässä on ongelmana paitsi ominaisuuksiltaan rajoittuneempien päätelaitteiden ja hitaiden verkkojen hyödyntäminen myös ylläpito, sillä useiden versioiden ylläpitäminen samasta palvelusta voi olla työlästä.

Princess-projektissa kehitetään alustaa, joka helpottaa palvelujen adaptointia eri päätelaitteisiin. Projektin tutkimusosapuolina ovat VTT Elektroniikka (http://www.ele.vtt.fi) ja Oulun yliopiston Mediateam (http://www.mediateam.oulu.fi). Projektia rahoittavat Tekes, eräät yritykset sekä tutkijaosapuolet itse. VTT:n osuutena projektissa on kehittää ratkaisuja sessionhallintaan, palvelun käyttöä tukevaan älykkyyteen (esimerkiksi käyttäjien profilointi) sekä käyttöliittymän esittämiseen eri alustoilla (HTML, Java, WAP, SMS tai puheohjaus). Yliopisto tekee projektissa median skaalausta (kuva, video, teksti jne.) sekä palvelurajapintoja toteuttavia osia.

Tässä tutkielmassa käytettävät käsitteet voi määritellä seuraavasti: Käyttäjä haluaa tietoa tai ajanvietettä, jota voidaan kutsua *sisällöksi*. Sisällön esitysmuotoa (toisin sanottuna ilmaisuvälinettä) voidaan kutsua *mediaksi* tai *esitykseksi*. *Mediatyyppejä* ovat esimerkiksi teksti, muotoiltu teksti, kuvat, video ja ääni, joista kolme viimeistä ovat *binäärimuotoisia*. *Palveluksi* kutsutaan sitä kokonaisuutta, joka luo sisällön ja sen esityksen. Sisällön siirtämiseksi käyttäjälle tarvitaan tietty *toiminnallisuus* ("moottori"), jolla haluttu esitys muodostetaan. Toiminnallisuuden ohjaamiseen tarvitaan *käyttöliittymä* ("hallintalaitteet").

Tutkimusongelma

Mitä haasteita käyttöliittymän adaptointi eri päätelaitteisiin ja verkkoihin tuo, ja miten niihin voidaan vastata?

Edellä mainittua kysymystä voidaan tarkentaa seuraavasti:

1. Millaisia käyttöliittymäratkaisuja ja palvelutyyppejä eri päätelaitteet tukevat?

2. Mikä on adaptiivinen käyttöliittymä, ja miten käyttöliittymää voidaan adaptoida?

3. Miten adaptoiva järjestelmä voidaan rakentaa, ja mitä työkaluja voidaan käyttää?

Näillä kysymyksillä tutkielmassa pyritään lisäämään tietämystä eri päätelaitteista ja verkoista käyttöliittymien näkökulmasta ja siitä, miten tätä moninaisuutta voidaan hallita. Aiheen rajauksena toimii pitkälti se, mitä projektissa tutkitaan ja mikä on sille hyödyksi. Toisaalta kaikki projektissa tutkittava ei kuulu tähän tutkielmaan, vaikka niiden aiheuttamat vaatimukset täytyykin jossain määrin ottaa huomioon:

- yleinen sessionhallinta ja käyttäjän tietojen hallinta
- laskutus ja käyttäjän profilointi, sekä miten näiden tulisi näkyä käyttäjälle
- kuvien, videon yms. median adaptointi eri päätelaitteisiin sopivaksi
- palvelurajapinta ja palvelujen toteutus.

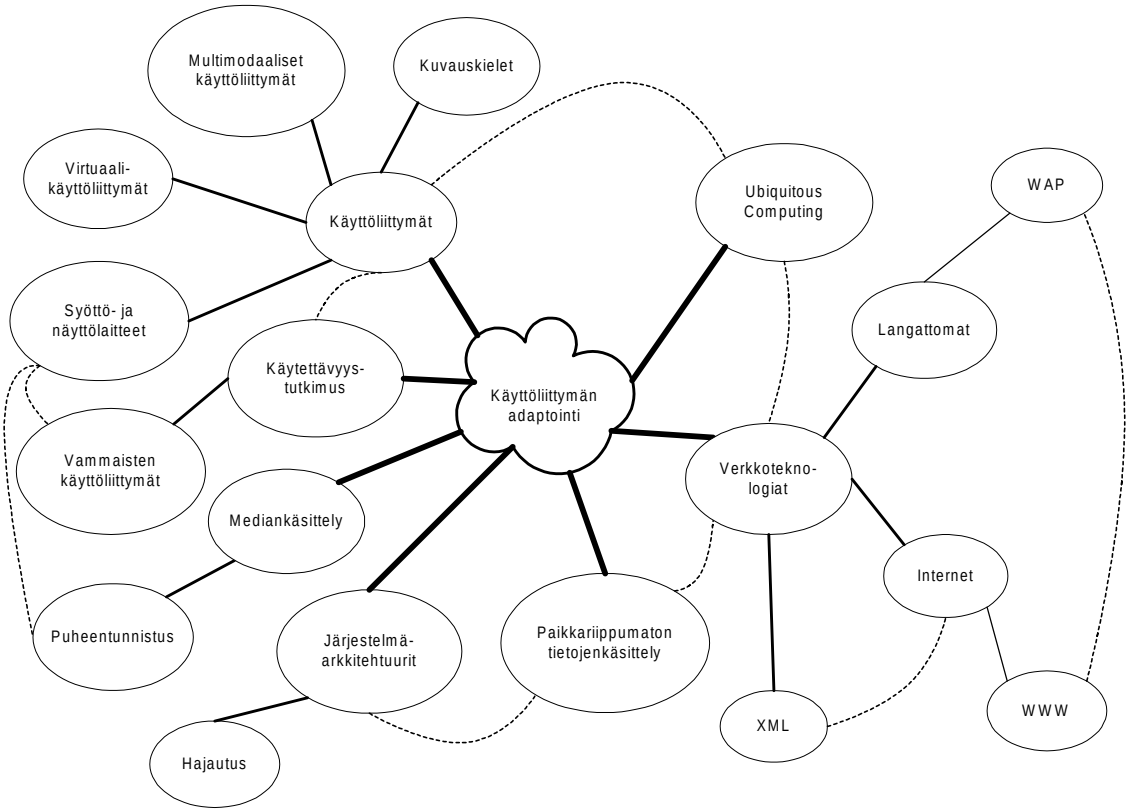
Tutkimusmenetelmä

Tutkimusmenetelmä on keinoja painottava konstruktiivinen tutkimus: lopputilanne on osittain määritelty ylemmällä tasolla jo projektia haettaessa, joten tutkimuksessa yritetään etsiä ja kehittää ratkaisuja siihen pääsemiseksi (Järvinen & Järvinen 1996).

Lähdeaineisto

Lähdeaineisto koostuu käyttöliittymäkirjallisuudesta, teknisistä standardeista ja dokumenteista sekä Princess-projektissa tuotetusta materiaalista.

Kirjallisuus on lähteenä selvitettäessä sitä, mitä käyttöliittymät ovat, minkälaisia käyttöliittymiä on eri päätelaitteissa sekä miten ne on toteutettu. Tutkielman kannalta mielenkiintoinen alue ovat vammaisille suunnitellut apuvälineet ja -ohjelmat, sillä Vanderheidenin (1997) mukaan esimerkiksi sokealle käyttäjälle suunnitellut ratkaisut ovat käyttökelpoisia myös tilanteessa, jossa päätelaitteessa on vain pieni näyttö tai ei näyttöä ollenkaan. Sama koskee myös liikuntarajoitteisia ja laitteita, joissa on pieni näppäimistö tai ei näppäimistöä ollenkaan.



Kuva 2. Läheisiä tutkimusalueita.

Tutkielmaan liittyy paljon tulossa olevia tai muualla tutkimuksen kohteena olevia tekniikoita, joista edellä olevassa kuvassa ovat ainakin kaikkialla läsnä oleva (ubiquitous) ja paikkariippumaton (nomadic) tietojenkäsittely sekä virtuaaliset ja multimodaaliset, useaan aistiin perustuvat käyttöliittymät. Nämä jäävät kuitenkin maininnan tasolle, sillä Princess-projekti perustuu olemassa olevien päätelaitteiden käyttöön.

Verkkoteknologiat liittyvät tutkielmaan sikäli, että protokollat vaikuttavat hajautetussa ympäristössä ainakin vuorovaikutuksen suuntaan sekä tiedonsiirtokaista ja luotettavuus interaktion laatuun ja määrään.

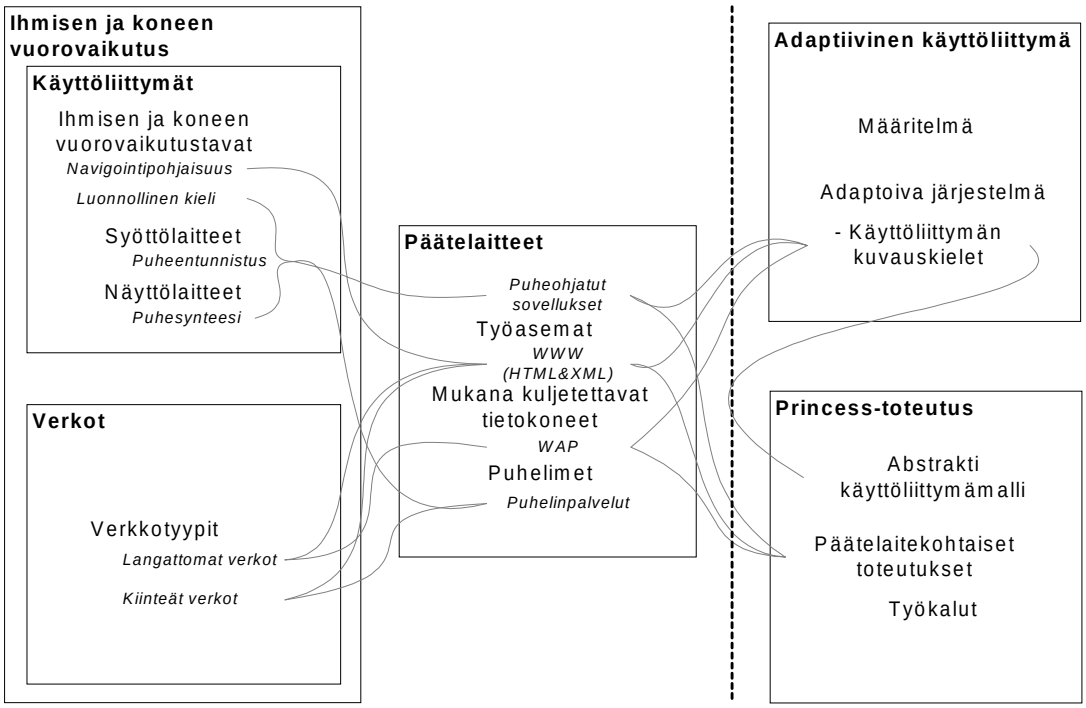
Tutkielmassa käsiteltävistä teknologioista on runsaasti tietoa erilaisten konsortioiden, yritysten ja standardointijärjestöjen kotisivuilla. Tutkielman kannalta mielenkiintoisimpia ovat W3C (World Wide Web Consortium, http://www.w3.org), Java (http://java.sun.com) ja WAP forum (http://www.wapforum.org).

Tutkielman tulokset ja rakenne

Tutkielma jakautuu (kuva 3) yleiseen osaan ja konstruktiiviseen, Princess-spesifiseen osaan.

Yleisen osan alussa käydään läpi ihmisen ja koneen vuorovaikutusta sekä siihen liittyviä laitteita. Tämän jälkeen kuvataan lyhyesti erilaisia verkkoteknologioita. Kolmannessa luvussa käydään läpi olemassa olevia päätelaitetyyppejä, käyttöliittymiä ja verkkoja.

Tutkielman konstruktiivisen osan alussa määritellään tarkemmin adaptiivinen käyttöliittymä ja se, miten käyttöliittymää voi adaptoida. Toteutukseen painottuvan osan lähtökohtana on projektin ensimmäisen vaiheen prototyyppi, jonka ylläpitoon ja osin myös toteutukseen olen osallistunut. Tutkielmassa kuvataan lyhyesti tätä toteutusta. Pääasia on kuitenkin ensimmäisen vaiheen kokemuksien huomioonottaminen ja sen perusteella tapahtuva toisen vaiheen suunnittelu ja toteutus.



Kuva 3. Tutkielman rakenne. Kursiivilla painetut osat ovat esimerkkejä tutkielmaan liittyvistä asioista.

2. Ihmisen ja koneen vuorovaikutus

Tässä luvussa käsitellään yleisesti ihmisen ja koneen vuorovaikutusta ja siihen vaikuttavia tekijöitä. Luku ei vastaa tutkimusongelmiin vaan määrittelee myöhemmissä luvuissa käytettäviä käsitteitä.

2.1 Käyttöliittymät

Mikä on käyttöliittymä? Alla olevat määritelmät on koottu lähteestä Preece (1994, s. 7).

> *those aspects of a system that the user comes in contact with – T.P. Moran, 1981.*

tai

> *an input language for the user, an output language for the machine, and a protocol for the interaction – H.U. Chi, 1985.*

Käyttöliittymä-käsitteen rajoite on sen suppeus: se huomioi pelkästään sen, mitä on ihmisen ja koneen välissä. Laajemmin täytyisi ottaa huomioon myös ulkopuoliset vuorovaikutukseen liittyvät asiat sekä se, mitä tapahtuu ihmisessä ja koneessa itsessään. Tällainen käsite on ihmisen ja koneen vuorovaikutus (HCI). HCI voidaan määritellä seuraavasti:

> *[a] set of processes, dialogues and actions through which a human user employs and interacts with a computer – R.M. Baecker & W.A.S Buxton, 1987.*

tai laajemmin

> *Human-computer interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them. – ACM SIGCHI, 1992.*

Viimeisin määritelmä on jopa niin laaja, että on vaikeaa tai mahdotonta rajata käyttöliittymää mitenkään. Järjestelmää ei voi suunnitella ottamatta huomioon sen käyttöliittymää ja päinvastoin: ei voi suunnitella käyttöliittymää ottamatta huomioon muuta systeemiä.

Tämän tutkielman aihe, adaptiivisuus, on käyttöliittymän piirre, joka mittaa yleisesti sitä, kuinka hyvin käyttöliittymä sopeutuu tai voidaan sopeuttaa olosuhteisiin, käyttäjän tarpeisiin tai muuhun vastaavaan tekijään (luku 4).

2.1.1 Käytettävyys

Preecen (1994, s. 722) mukaan käytettävyys mittaa helppoutta, millä järjestelmä voidaan oppia tai sitä voidaan käyttää, sekä sen turvallisuutta, tarkoituksenmukaisuutta, tehokkuutta ja asennetta käyttäjää kohtaan. Princess-järjestelmän tarkoitus voidaan määritellä myös pyrkimyksenä tehdä verkkopalvelut käytettäviksi eri päätelaitteiden ja verkkojen kautta.

Käytännönläheinen lähestymistapa käytettävyyteen ovat heuristiikat. Ne ovat suhteellisen yleisesti hyväksyttyjä periaatteita, joita Nielsenin (1993, s. 20) mukaan "jokaisen käyttöliittymän suunnittelijan tulisi noudattaa". Taulukossa 1 on listattu kymmenen Nielsenin heuristiikkaa.

Taulukko 1. Nielsenin (1993, 20) käytettävyysheuristiikat.

1	*Yksinkertainen ja luonnollinen dialogi*	Järjestelmän pitää kätkeä epäoleelliset seikat ja esittää oleelliset luonnollisessa ja loogisessa järjestyksessä.
2	*Puhu käyttäjän kieltä*	Kielen ja käsitteiden pitää olla käyttäjälle tuttuja.
3	*Älä kuormita käyttäjän muistia liikaa*	Muistettavien asioiden määrä pitää minimoida. Käyttöohjeiden täytyy olla helposti saavutettavissa.
4	*Konsistenssi*	Järjestelmän termistön ja käyttäytymisen täytyy vastata toisiaan eri tilanteissa.
5	*Palaute*	Käyttäjän täytyy tietää, mitä järjestelmässä tapahtuu.
6	*Poistumistiet*	Virheellinen valinta pitää olla peruttavissa.
7	*Oikotiet*	Kokeneet käyttäjät tarvitsevat nopeita oikoteitä.
8	*Hyvät virheilmoitukset*	Hyvä virheilmoitus kertoo sanallisesti, mikä aiheutti virheen ja miten ongelma ratkaistaan.
9	*Virheiden estäminen*	Tämä on hyviä virheilmoituksia parempi vaihtoehto.
10	*Apu ja dokumentaatio*	Ohjeiden pitää olla konkreettiset, sopivan laajat, kohdistettu käyttötilanteeseen ja helposti löydettävissä.

2.1.2 Vuorovaikutustavat

Vuorovaikutustavat ovat tapoja, joilla kone ja ihminen keskustelevat. Tässä luvussa esitetty lista on Preecen (1994, s. 262–) esittämä. Luvussa 2.1.3 käsitellään tarkemmin niitä laitteita ja teknologioita, joita tarvitaan vuorovaikutuksen toteuttamiseen.

Komentopohjainen tapa

Komentopohjaisuus perustuu funktionäppäinten käyttöön tai näppäimistöltä annettaviin komentosanoihin tai niiden lyhenteisiin.

Valikko- ja navigointipohjainen tapa

Valikko- ja navigointipohjaisuus vähentää käyttäjän muistin kuormitusta, sillä hän näkee näytöllä ohjaamiseen käytettävät elementit sen sijaan, että niitä tarvitsee erikseen muistaa.

Lomake- ja taulukkopohjainen tapa

Tämä tapa vastaa paperilla olevia lomakkeita, joten se soveltuu parhaiten datan syöttämiseen.

Luonnollinen kieli

Luonnollinen kieli kirjoitettuna tai puhuttuna on käyttäjälle helppo mutta tietotekniikan näkökulmasta hankala. Sen käsittelyä vaikeuttavat kielen moniselitteisyys, epämääräisyys, käyttäjien kielioppivirheet sekä erilaiset käyttötavat, kuten murteet. Toimivimmat ratkaisut perustuvatkin usein rajoitettuun sanastoon ja kielioppiin. Toisaalta se ei ole myöskään aina tehokkain tapa: esimerkiksi hitaalle kirjoittajalle valikkopohjainen tapa voi olla kirjoitettua tekstiä parempi tai vastaavasti kokenut käyttäjä hyötyy minimaalisesta komentokielestä. (Preece 1994, s. 269.)

Pelkästään puheeseen perustuvissa käyttöliittymissä tarvitaan alla lueteltuja teknologioita (Kamm et al. 1997). Vastaavia ongelmia joudutaan ratkomaan myös kirjoitetun tekstin yhteydessä.

- *Puheen ja äänen koodaus*, mikä mahdollistaa tiedon siirron ja tallennuksen toisaalta tehokkaasti mutta toisaalta niin, että äänen laatu säilyy riittävän hyvänä.

- *Puhesynteesi ja puheentunnistus* eli tietokoneen suu ja korvat.

- *Kielen ymmärtäminen* eli tietokoneen aivot, jonka avulla luodaan merkityksiä puheentunnistuksen tuottamalle datalle.

- *Työkalut ja kuvauskielet*, joiden avulla voidaan suunnitella ja toteuttaa dialogi tietokoneen ja ihmisen välille.

Suora manipulointi

Suora manipulointi perustuu kohteen tai sitä esittävän ikonin suoraan käsittelyyn. Esimerkkejä tästä ovat piirto-ohjelmat, pelit ja työpöytä-metaforaan perustuvat käyttöliittymät.

2.1.3 Laitteet

Ihmisen ja koneen vuorovaikutuksessa käytettävien laitteiden täytyy vastata käyttäjän fysiologisia ja psykologisia piirteitä sekä olla soveliaita kuhunkin tehtävään ja ympäristöön (Preece et al. 1994, s. 204). Vaatimukset saattavat kuitenkin olla ristiriitaisia – esimerkiksi langattomaan ympäristöön ei välttämättä saa tehtävään sopivia laitteita, joten käytännössä joudutaan tyytymään kompromisseihin.

2.1.3.1 Syöttölaitteet

Syöttölaitteiden avulla käyttäjä voi muuntaa informaatiota koneella käsiteltävään muotoon (Preece et al. 1994, s. 212). Näitä laitteita ovat esimerkiksi erilaiset näppäimistöt, puheentunnistimet ja osoitinlaitteet, kuten hiiret, ohjaussauvat, kosketusnäytöt ja kynät.

Luonnollisen kielen käyttöä ja sen ongelmia on käsitelty aiemmin vuorovaikutustapojen yhteydessä. Luonnolliseen kieleen liittyvät syöttölaitteita voivat olla näppäimistö, kynä tai puheentunnistin. Näppäimistö on näistä järjestelmän kannalta helpoin, sillä esimerkiksi kynää käytettäessä eri käsialojen tunnistaminen on hankalaa; joidenkin henkilöiden kirjoitus on paremmin tunnistettavaa kuin toisten (Preece et al. 1994, s. 234).

Puheentunnistuksen ongelmia ovat äännetyn ja kirjoitetun kielen eroavuus, puhesignaalin vaihtelevuus sekä ääniväylän ja ulkoisen melun aiheuttamat häiriöt. Lisäksi joissakin tapauksissa rajoittavana tekijänä voivat olla tunnistimen prosessoriteho- ja muistivaatimukset. Osa puheentunnistuksen ongelmista on ratkaisematta, joten tulosta pyritään parantamaan parametroimalla puheentunnistin eri tavalla eri käyttökohteisiin. Käytettäviä parametreja ovat sanavaraston koko, tunnistettava ääniyksikkö, puheen jatkuvuus ja puhujariippumattomuus. Ideaalinen puheentunnistin sisältäisi koko luonnollisen kielen sanaston, olisi puhujariippumattomaton ja tunnistaisi ymmärrettävästi äännettyä, jatkuvaa puhetta. Käytännössä yksinkertaisimmat tunnistimet tunnistavat kokonaisia sanoja kiinteästi määritellystä ja suppeasta sanavarastosta – monimutkaisemmissa tunnistaminen tapahtuu äänteiden tai äännekombinaatioiden perusteella vaihtuvasta tai suuresta sanavarastosta. (Peltola 1998; Rabiner & Juan 1993.)

Ääneen liittyy myös useita alla listattuja ominaisuuksia, jotka erottavat sen muista vuorovaikutuksen välineistä (Grasso et al. 1998):

- Tilapäinen luonne – ääni ei jää "näkyviin".
- Äänen käyttö ei vaadi fyysistä kosketusta, joten sitä voidaan käyttää kohtuullisen etäisyyden päästäkin.
- Ääni on ympärisäteilevää, joten sen suuntaaminen tiettyyn kohteeseen voi olla vaikeaa.
- Äänellä ohjattavan järjestelmän kyvyt yliarvioidaan herkästi, sillä sitä saatetaan pitää todellista ihmismäisempänä.

Erityisesti virtuaalitekniikoissa käytetään myös eri aistikanaviin perustuvia sensoreita: esimerkiksi kehon liikettä ja paikkaa voidaan havainnoida kolmiulotteisilla paikkasensoreilla tai käden liikettä datahansikkailla (Burdea & Coiffet 1994, s. 15–45). Näiden lisäksi on olemassa ryhmä syöttölaitteita, joita käyttäjä ei ohjaa suoraan vaan jotka mittaavat käyttötilannetta. Todettavia asioita voivat olla laitteen sijainti, asento, ympäröivät fyysiset olosuhteet ja infrastruktuuri tai käyttäjän ominaisuudet ja toimenpiteet. Näiden mittaustietojen hyväksikäyttöä kutsutaan kontekstiherkkyydeksi (context awareness) – toisin sanoen laite ja ohjelmisto sopeutuvat käyttötilanteeseen. Esimerkkisovelluksia voivat olla esimerkiksi puhelimen hälytysäänen automaattinen hiljeneminen kokouksissa, unohtuneista asioista muistuttava matkapuhelin tai käsialantunnistin, jonka tarkkuutta parannetaan ottamalla huomioon, onko käyttäjä paikallaan vai liikkuvassa kulkuneuvossa. Sovellukset voivat käyttää laitteisiin upotettuja mittalaitteita tai yleistä infrastruktuuria, kuten GPS-paikannussatelliitteja tai GSM-verkosta saatavia tietoja. (Schmidt et al. 1998; Brown et al. 1997.)

2.1.3.2 Näyttölaitteet

Näyttölaite on tässä yhteydessä hieman huono termi, sillä se viittaa visuaaliseen esitykseen. Yleisemmin kyseessä ovat kaikki välineet, joiden kautta järjestelmä voi viestiä käyttäjälle.

Yleisimpiä visuaalisuuteen perustuvia laitteita ovat kaksiulotteiset näytöt tai tulostimet. Niiden käytössä täytyy ottaa huomioon kolme seikkaa (Preece et al. 1994, s. 239): fyysiset tekijät (esimerkiksi kirkkaus, värikombinaatiot), tapa, jolla informaatio esitetään (tekstin koko, elementtien järjestely) ja tapa, jolla informaatiota käytetään.

Äänet, olivatpa ne sitten puhetta, musiikkia tai äänimerkkejä, ovat myös yleinen käyttöliittymän esitysmuoto. Ne sopivat tilanteisiin, joissa ei joko käyttäjän, käyttötilanteen tai muiden syiden takia voida käyttää visuaalista esitysmuotoa. Niitä voidaan käyttää myös täydentämään muita esitysmuotoja esimerkiksi siten, että äänimerkin avulla kiinnitetään käyttäjän huomio virhetilanteessa sellaiseen kohteeseen, jota hän ei tavallisesti seuraa. Äänen tuottaminen ei ole teknisesti vaikeaa synteettistä puheen generointia lukuun ottamatta, jossa ongelmana on luonnollisuuden saavuttaminen. (Preece et al. 1994, s. 247)

Virtuaalitekniikoissa käytetään monipuolisesti hyväksi näkö-, kuulo- ja tuntoaisteja. Kuva voidaan luoda projisoimalla se suoraan silmään, käyttämällä silmien eteen asetettavia näyttöjä (HMD) tai stereolasien ja tavallisen näytön avulla. Mekaanisten laitteiden synnyttämän paineen tai värinän avulla voidaan hyödyntää tuntoaistia, minkä lisäksi voidaan luoda myös fyysinen vastus (force feedback). (Burdea & Coiffet 1994, 45–.)

2.2 Verkot

Verkkojen ja niiden protokollien asettamat rajoitukset vaikuttavat ihmisen ja koneen vuorovaikutukseen erityisesti hajautetuissa järjestelmissä. Ne voivat esimerkiksi vaikuttaa interaktion suuntaan, nopeuteen, helppouteen tai interaktiossa käytettyihin mediatyyppeihin. Tässä luvussa käsitellään näitä tekijöitä lähteiden CTI (1999) ja Forman & Zahojan (1994) pohjalta.

Yhteyden taso

Yhteyden tason (quality of service) parametreja ovat

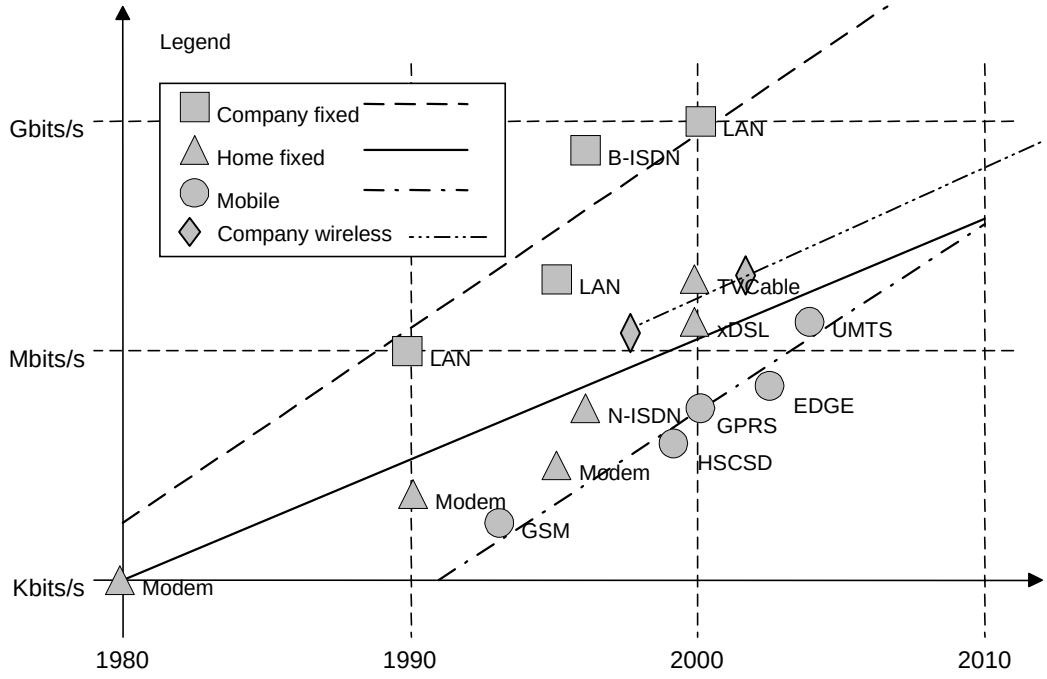
- kaistanleveys ja sen vaihtelut. Kaistanleveys voidaan mitata joko käyttäjäkohtaisesti tai kokonaismääränä, jonka käyttäjät jakavat keskenään.

- vasteajat sekä yhteyden muodostamisessa että käytön aikana

- virheiden aiheuttajat, määrä ja niistä toipuminen.

Nämä tekijät vaikuttavat käyttömukavuuteen tai pahemmassa tapauksessa rajoittavat käyttöä. Esimerkiksi pieni kaistanleveys ei salli raskaita multimediaesityksiä tai pitkät vasteajat eivät sovi reaaliaikaista vuorovaikutusta vaativiin sovelluksiin. Järjestelmän käytettävyyteen vaikuttaakin se, kuinka hyvin se sopeutuu näihin tekijöihin käyttäjälle läpinäkyvästi.

Langattomat verkot ovat yhteyden laadun suhteen yleensä kehittymättömämpiä kuin langalliset. Niiden ominaisuudet kuitenkin kehittyvät voimakkaasti (kuva 4): uudet verkot, kuten GPRS, tarjoavat suuremman kaistanleveyden sekä pakettipohjaisen tiedonsiirron, jonka avulla voidaan pienentää yhteyden muodostamisesta aiheutuvia viipeitä.



Kuva 4. Eri verkkotyypit ja niiden kaistanleveys (CTI 1999).

Tietoturva

Tietoturvan kolme peruskäsitettä ovat luottamuksellisuus, eheys ja käytettävyys. Luottamuksellisuuteen kuuluu se, että tieto on saatavilla vain niille henkilöille, joille se on tarkoitettu. Eheys merkitsee sitä, että tieto ei muutu varastoinnin tai siirron aikana tarkoituksettomasti. Tiedon on oltava myös käytettävissä missä ja milloin käyttäjät sitä tarvitsevatkin.

Tietoliikenteen tietoturvassa tarvittavat teknologiat liittyvät käyttäjän tunnistukseen, siirron ja tallenteen salaukseen ja asiattomalta käytöltä suojautumiseen. Tietoturvan peruskäsitteet saattavat olla käytännössä ristiriitaisia – esimerkiksi luotettava tunnistus ja salaus voivat haitata tiedon käytettävyyttä. Käytännön sovelluksissa joudutaankin turvautumaan riskianalyysiin ja valitsemaan sen pohjalta riittävä turvallisuustaso. Valintoja rajoittaa myös kunkin verkon tuki esimerkiksi käyttäjän tunnistamiseen.

Liikkuvuuden tuki

Käyttäjä voi liikkua joko päätelaitteesta toiseen tai päätelaitteen kanssa. Ensimmäisessä tapauksessa käyttäjä käyttää kulloinkin eteen tulevia päätelaitteita. Tällöin oleellisia seikkoja ovat tunnistaminen ja käyttäjäkohtaisten asetusten siirtyminen paikasta toiseen. Jälkimmäisessä tapauksessa oleellinen tekijä on tiedonsiirtokanava: Lankaverkoissa päätelaitteen liikkuvuus on luonnollisesti huono. Langattomissa verkoissa taas on merkityksellistä, käytetäänkö esimerkiksi infrapuna- vai radiosiirtoa sekä mikä on käytettävä taajuus ja verkon solun koko. Käytettävät protokollat vaikuttavat siihen, kuinka läpinäkyvästi päätelaite osaa siirtyä paikasta (eli solusta) toiseen.

Hinta ja laskutustapa

Verkkoihin liittyy monia laskutustapoja ja -perusteita: yleisesti käytetään kiinteää hintaa (esimerkiksi kuukausimaksu), palvelussa käytettyä aikaa tai siirretyn tiedon määrää. Käyttäjälle oleellisia seikkoja ovat paitsi laskun suuruus myös laskutuksen läpinäkyvyys ja tietoturva eli laskutuksen luotettavuus. Läpinäkyvyydellä tarkoitetaan tässä sitä, kuinka käyttäjä näkee laskutuksen: toisaalta se ei saa haitata palvelun käyttöä olemalla esillä liian näkyvästi, mutta toisaalta käyttäjän täytyisi tietää palvelun hinta.

3. Päätelaitteiden käyttöliittymät ja verkot

Tässä luvussa esitellään lyhyesti eri päätelaitetyyppejä. Käsiteltäviä asioita ovat niihin sopivat käyttöliittymätyypit, mahdolliset ongelmat, tyypillisten verkkojen vaikutukset sekä käyttöliittymien toteutukseen liittyvät kuvauskielet ja muut toteutusteknologiat adaptiivisuuden näkökulmasta.

3.1 Työasemat

Työasema on käsitetty tässä laajasti: se voi olla PC, verkkotietokone (NC), X-pääte tai muu vastaava. Sille on yleensä varattu paikka työpöydällä, joten käytettävissä on täysikokoinen näppäimistö, hiiri tai vastaava osoitinväline, kuva-alaltaan suuri monitori, mahdollisesti myös multimediaominaisuudet, riittävästi prosessoritehoa sekä mahdollisuus käyttää nopeita verkkoja. Työasemien käyttöliittymissä toimivat kaikki vuorovaikutustavat; luonnollisen kielen käyttö tosin rajoituksin (Preece et al. 1994, 261–).

Työasemakäyttöliittymät on tässä jaettu kahteen ryhmään, "perinteisiin" työpöytäsovelluksiin ja selainpohjaisiin. Ryhmät erottaa toisistaan se, että ensin mainitussa tapauksessa käyttöliittymän hallinta tapahtuu kokonaan työasemassa, jälkimmäisessä tapauksessa se on osin hajautettu palvelimeen.

3.1.1 Työpöytäsovellukset

Työpöytäsovellukset suunnitellaan ja ohjelmoidaan usein tiettyyn laiteympäristöön ja käyttöjärjestelmään, jonka jälkeen ne käännetään valitun ympäristön konekielelle. Toisaalta on olemassa myös ajon aikana tulkattavia kieliä, joiden avulla voidaan nostaa ohjelmoinnin abstraktiotaso käyttöjärjestelmä- ja laitteistoriippumattomaksi.

Käännettävät ohjelmat ja niiden käyttöliittymät perustuvat joko laitteiston tai käyttöjärjestelmän suoraan käsittelyyn tai ylemmän tason komponenttikirjastojen käyttöön. Ensin mainittu vaihtoehto sitoo toteutukseen valitun ympäristön erityispiirteisiin, joten siirto muihin ympäristöihin on hankalaa. Toisaalta komponenttikirjastotkaan eivät usein toimi kuin yhdessä ympäristössä ja samankaan ympäristön eri kirjastot eivät ole yhteensopivia. Tällaisten sovellusten sovittaminen muihin ympäristöihin tapahtuukin suunnittelu- ja toteutustasolla joko käyttämällä alustariippumattomia kirjastoja tai toteuttamalla käyttöliittymä erikseen jokaiselle alustalle. Toteutuksen jälkeinen alustalta toiseen siirtyminen on mahdollista ainoastaan emuloimalla alkuperäistä ympäristöä ohjelmallisesti.

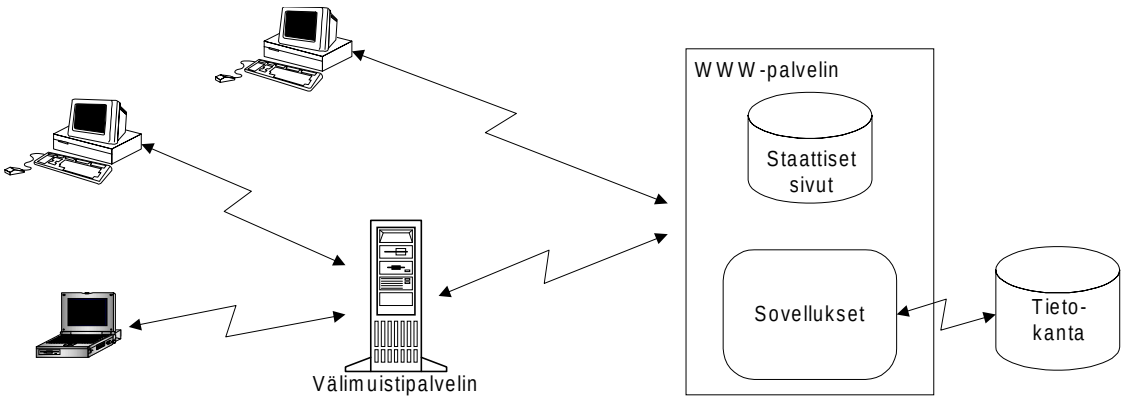
Java-teknologia määrittelee virtuaalikoneen, joka tarjoaa ohjelmille samat palvelut riippumatta todellisesta laiteympäristöstä ja käyttöjärjestelmästä. Java AWT-käyttöliittymäkirjasto määrittelee ne komponentit ja rutiinit, jotka ovat pienin yhteinen nimittäjä yleisimmissä graafisissa käyttöympäristössä. Swing, josta käytetään myös nimeä JFC, on AWT:n yläpuolelle Javalla rakennettu luokkakirjasto, joten sen avulla voidaan paremmin toteuttaa piirteitä, joita ei välttämättä löydy kaikista ympäristöistä. Sekä AWT että Swing tekevät käyttöliittymästä siirrettävän niille alustoille, joille Java-virtuaalikone on toteutettu ja joissa on graafinen näyttö – ongelmia tosin voivat aiheuttaa esimerkiksi erilaiset näyttöjen koot tai ohjelman suorituskyky.

Muita alustariippumattomia tekniikoita ovat ainakin Tcl-ohjelmointikieli ja siihen liittyvä Tk-käyttöliittymäkirjasto sekä LISP-ohjelmointikieleen perustuva CLIM. CLIM:n ominaisuuksiin kuuluu muun muassa se, että käyttöliittymäkomponenttien sijasta käytetään ylemmän tason abstraktiota, jolloin käyttöliittymänhallintajärjestelmä valitsee itse tilanteeseen sopivan toteutuksen (McKay 1991).

Ohjelmistot suunnitellaan ja toteutetaan usein käytettäväksi ennalta määrätyillä laitteilla, joista yleisimmät ovat näyttö, näppäimistö ja hiiri. Myös ohjelmien ulkoasu, kuten käyttöliittymäkomponenttien koot, määrätään toteutusvaiheessa, joten alusta- ja esitystapariippumattomuus ei ole erityisen hyvä. Ongelmaan on kuitenkin etsitty ratkaisuja erityisryhmille, kuten vammaisille. Yleensä niissä apusovellus seuraa ja ohjaa käyttöjärjestelmän tarjoamien rajapintojen kautta sovelluksen toimintaa. Esimerkiksi Edwardsin ja Mynattin (1994) X Window -ympäristöön toteuttama järjestelmä muodostaa ohjelman käyttöliittymästä puumaisen hierarkian, jossa käyttäjä voi navigoida puhesyntetisaattorin ja muiden äänien perusteella näppäimistön tai puheentunnistuksen kautta. Vastaavia järjestelmiä on olemassa myös Windows-(Microsoft Accessibility-- 1999) ja Java-ympäristöön (Schwerdtfeger 1998).

3.1.2 WWW-selainsovellukset

Selainsovellukset perustuvat monitasoarkkitehtuuriin, jossa käyttöliittymä, sovelluslogiikka ja tiedon tallennus on erotettu toisistaan (kuva 5). Tunnetuin ja yleisimmin käytetty esimerkki tästä on WWW, jota tämä kappale käsittelee.



Kuva 5. WWW-selainsovellusten arkkitehtuuri.

WWW:n perusperiaate on yleismaailmallisuus, joka tarkoittaa riippumattomuutta laite- tai ohjelmistoalustasta, verkkoinfrastruktuurista, kielestä, kulttuurista, maantieteellisestä sijainnista tai käyttäjän henkisestä tai fyysisestä vajavaisuudesta. Tekijöitä, joilla tähän pyritään, ovat universaali resurssien nimeämiskäytäntö (URI, universal resource identifier) sekä yhteiset protokollat ja sisällönkuvauskielet. (Berners-Lee, 1998.)

WWW:ssä käytetyistä sisällönkuvauskielistä tärkein on HTML. Sen perusperiaatteita on sisällön ja esitystavan erottaminen toisistaan, mikä on tärkeää pyrittäessä alustariippumattomuuteen. (Berners-Lee 1998.) Ongelmia vain on aiheuttanut se, että alunperin ei ollut keinoja, joilla palveluntarjoaja olisi voinut vaikuttaa esitystapaan. Tästä seurasi se, että ulkoasua korostava koulukunta alkoi kehittää HTML-laajennoksia, jotka mahdollistavat graafisen suunnittelun käyttämällä mainonnasta, multimediaesityksistä ja paperijulkaisuista tuttuja keinoja: muotoja, värejä, typografiaa sekä grafiikan ja tekstin asettelua. Tämän seurauksena esimerkiksi bittikarttakuvia käytetään muotoillun tekstin esittämiseen ja toisaalta HTML-elementtejä käytetään semanttisesti väärin: taulukkoja käytetään usein tekstin ja grafiikan tarkkaan sijoittamiseen ruudulle tai otsikot luodaan typografisin keinoin semanttisten elementtien sijasta, katso esimerkiksi Lynch & Horton (1999).

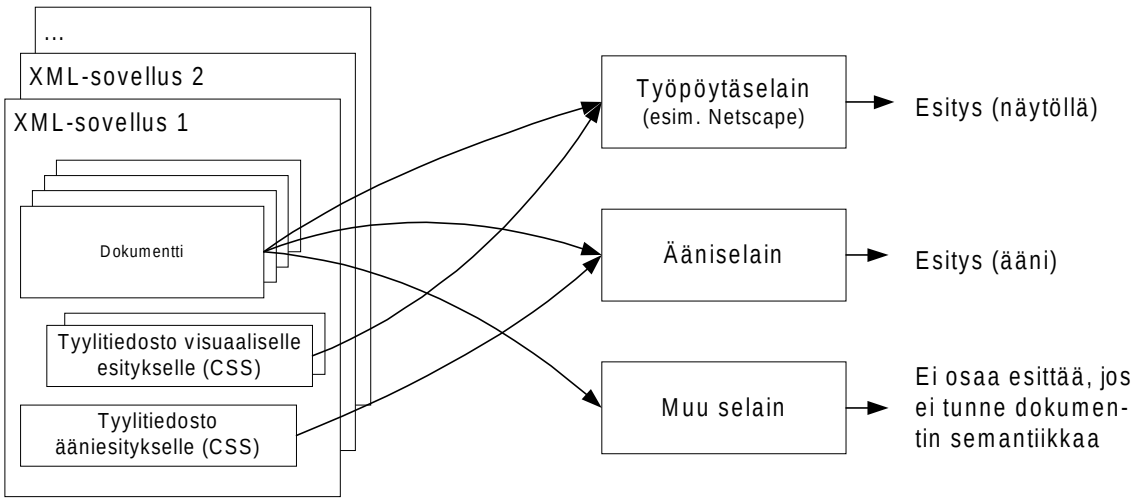
Parempi keino ulkoasun määrittämiseen ovat CSS-tyylitiedostot. Yhtä dokumenttia varten voidaan määrittää useita tyylitiedostoja: Palvelun kehittäjä voi määritellä useita visuaalisia tai äänellisiä tyylejä eri käyttäjäryhmille sekä laite- ja ohjelmistoalustoille. Lisäksi käyttäjällä voi olla omia asetuksia, jotka ohittavat tarvittaessa palvelun asetukset. Viimeksi mainitusta syystä CSS on erityisen käyttökelpoinen niille käyttäjäryhmille, kuten näkövammaisille, joita ei tavallisesti oteta huomioon palveluita kehitettäessä. (W3C 1998a.) CSS:n yleistymistä ovat rajoittaneet huono tuki selaimissa ja kehitystyökaluissa sekä eräät puutteet syntaksissa (Marden & Munson 1999).

Uusi WWW-teknologia on XML. Se on edeltäjäänsä SGML:ää yksinkertaisempi tapa kehittää omia dokumenttityyppejä eli sovelluksia. Lisäksi sen tavoitteita ovat dokumen-

tin käsittely osittaisen ymmärtämisen periaatteella (toisin sanoen ei haittaa, vaikka dokumentissa on sellaisia osia, joiden käsittelyä ei ole toteutettu käytössä olevassa työkalussa) sekä mahdollisuus sekoittaa keskenään eri dokumenttityyppejä. (Berners-Lee 1998.) Koska XML on metakieli, sille voidaan kehittää rajaton määrä sovelluksia eri käyttötarkoituksiin (rakenteellisten kielten yhteydessä kannattaa huomata, että sovellus ei ole tässä yhteydessä sovellusohjelman synonyymi: esimerkiksi HTML on SGML:n sovellus, koska se on määritelty SGML:n mukaisesti ja dokumentit noudattavat SGML-syntaksia).

Koska XML:n sovelluksia voi määritellä itse, ei XML-dokumentilla ole kiinteästi määriteltyä semantiikkaa kuten HTML-dokumentilla. Niinpä dokumentti on oletusarvoisesti vain merkityksetön joukko elementtejä ja tekstiä: esimerkiksi <p>-elementti voi olla paketti, piano, pidike, tekstikappale (samoin kuin HTML:ssä) tai sitten jotain muuta. Semantiikan määrittelyyn ei ole olemassa yleispätevää formaalia tapaa, joten XML:n käsittely perustuu joko käsittelysääntöjen ohjelmointiin tai esitystavan määrittelyyn tyylitiedostoilla.

XML:n esittäminen tyylitiedoston avulla tarkoittaa käytännössä sitä, että jokaiselle sovellukselle tehdään vähintään yksi tyylitiedosto jokaista haluttua esitysmuotoa varten (kuva 6). Ongelmaksi voi muodostua se, että palvelujen kehittäjät eivät ota huomioon marginaalisina pitämiään käyttäjäryhmiä, kuten pienpäätelaitteiden ja ääniselainten käyttäjiä, ja sulkevat nämä siten pois.



Kuva 6. XML-dokumentin esittäminen.

Edellä mainitun perusteella paras tapa alustariippumattomaan dokumenttien esittämiseen XML-aikanakin olisi pitäytyminen yleisesti hyväksytyissä dokumenttityypeissä, jotka pystytään esittämään myös ilman tyylitiedostoja (Korpela 1998). Toisaalta tähän liittyy riski, että aiempi ulkoasun ja rakenteen ristiriita korvautuu uudella semanttisuu-

den ja alustariippumattomuuden ristiriidalla: toinen koulukunta pyrkii kehittämään eri käyttötarkoituksiin uusia dokumenttityyppejä, joihin on mahdollista soveltaa esimerkiksi nykyistä monipuolisempia hakumenetelmiä ja toinen pyrkii kehittämään yhtä alustariippumatonta kuvauskieltä, joka soveltuu useisiin käyttötarkoituksiin.

WWW-käyttöön sopivia standardoituja tai standardoitavia XML:n sovelluksia ovat ainakin MathML (matemaattiset kaavat), SVG (vektorigrafiikka), RDF (metatieto) ja XHTML (XML-syntaksin mukainen HTML) (Berners-Lee 1998). Erityisesti XHTML on päätelaiteriippumattomuuden kannalta mielenkiintoinen, sillä se modularisoi kuvauskielen sekä määrittelee menetelmät, joilla sitä voidaan laajentaa. Tämän avulla toisaalta päätelaitteet ja selaimet voivat ilmoittaa tukemansa moduulit tai toisaalta palvelut voivat ilmoittaa vaatimansa moduulit. Molempien vaatimusten tasapaino saavutetaan joko siten, että palvelin tai välimuisti (kuva 5) adaptoi palvelun päätelaitteen kykyjä vastaavaksi tai siten, että päätelaite lataa palvelun käyttämisessä tarvittavat ohjelmistot. (W3C 1999a.)

3.2 Mukana kuljetettavat tietokoneet

Mukana kuljetettavat tietokoneet ovat hajanainen joukko sekä ominaisuuksiltaan että käytetyltä termistöltä. Tässä luvussa käsitellään työasemia pienempiä laitteita, jotka ovat kulkevat mukana joko taskussa tai laukussa. Ryhmään kuuluvat niin kannettavat mikrot kuin taskumikrot ja kämmentietokoneetkin (PDA-laitteet).

Kannettavien laitteiden suunnittelun lähtökohta on usein ollut työaseman näytön, näppäimistön, virrankulutuksen, käyttöjärjestelmän tai työpöydän kutistaminen. Toisaalta joissain tapauksissa käyttöliittymän suunnittelussa on lähdetty soveltamaan vaihtoehtoisia tapoja, kuten luonnollista kieltä, kynäohjausta tai erikoiskäyttöjärjestelmiä tai -työpöytiä. Verkkotasolla käytetään yleensä samoja protokollia kuin työasemissakin, mutta toisaalta siirtotie on usein langaton.

Mukana kuljetettavissa tietokoneissa toimivat vastaavantyyppiset ohjelmat kuin työasemissakin: selainsovellukset, joissain tapauksissa Java-ohjelmat sekä "perinteiset", laitteisto- tai käyttöjärjestelmäriippuvaiset ohjelmat. Työasemissa ja niitä lähimpänä olevissa ratkaisuissa toimivat usein samat ohjelmat. Hankalammissa tapauksissa ohjelmia joudutaan muokkaamaan pienemmän näytön tai rajoittuneempien rajapintojen takia. Sama pätee myös selainsovelluksiin, jos niiden toiminta on suunniteltu päätelaitteen fyysisistä tai selaimen ominaisuuksista riippuvaisiksi (Yao 1999).

3.3 Puhelimet

Puhelimet ovat yleinen, suhteellisen halpa ja nykyään matkapuhelinteknologian ansiosta myös liikkuva päätelaitejoukko. Tässä luvussa käydään läpi kaikkiin puhelimiin soveltuva sovellustyyppi, äänisovellukset, sekä matkapuhelimissa toimivat tekstiviestit ja WAP-selainsovellukset.

3.3.1 Äänisovellukset

Puheeseen ja ääneen perustuva käyttöliittymä on luonnollinen valinta ja monesti myös ainoa mahdollisuus puhelimeen. Ääneen perustuvat järjestelmät ovat joko ihminen-kone tai ihminen-kone-ihminen -tyyppisiä (Kamm et al. 1997). Ensiksi mainitut ovat usein informaatiopalveluita (esimerkiksi aikataulut), joiden avulla käyttäjä etsii hakemansa tiedon itsepalveluna. Toisessa tyypissä kone on vain välittäjänä ihmisten välillä (esimerkiksi puheposti).

Puhelimen kautta toimivat järjestelmät voidaan luokitella sen mukaan, miten ohjaus tapahtuu (Kamm et al. 1997):

- *Äänitaajuus-ohjaus* (IVR) on vanhin ja teknisesti helpoimmin toteutettava tyyppi, jossa järjestelmää ohjataan puhelimen numeronäppäimillä. Tämä sopii paremmin kiinteän verkon puhelimiin kuin matkapuhelimiin, koska jälkimmäisissä näppäimistö on vuorovaikutteisuutta ajatellen hankalasti kuulokkeen kanssa samassa paketissa.

- *Puheohjaus* (SLI) voi perustua yksinkertaisiin, helposti tunnistettaviin puhekomentoihin tai luonnolliseen puheeseen. Puheentunnistuksen käyttöä voivat vaikeuttaa ympäristön melu sekä huono mikrofonin tai linjan laatu.

Toinen mahdollinen luokittelu perustuu järjestelmän antamaan palautteeseen (Kamm et al. 1997):

- *Nauhoitetut viestit,* joka ei sovi kovin dynaamisten palvelujen toteuttamiseen.

- *Puhetta sisältämättömät äänet* eli ääni-ikonit tai musiikki.

- *Synteettinen puhe,* jota voidaan generoida dynaamisesti tilanteen mukaan, mutta jonka ongelma on ihmisen puhetta heikompi ymmärrettävyys ja luonnollisuus.

Pelkkään ääneen perustuvien käyttöliittymien ongelma visuaalisiin käyttöliittymiin verrattuna on äänen lyhytkestoinen luonne yhdistettynä ihmisen huonoon muistiin (vrt. Nielsenin heuristiikka 3, s. 15). Kamm et al. (1997) ehdottavat seuraavia suunnittelukäytäntöjä:

- *Muistuttajat*, jotka auttavat käyttäjää, jos tämä ei järjestelmän mielestä edisty toivotulla tavalla (esimerkiksi on hiljaa liian kauan).

- *Konsistenssi* eri tilanteissa – esimerkiksi komentojen "peruuta" tai "apua" täytyy toimia joka tilanteessa.

- *Inkrementaalisuus*, jolloin järjestelmä ei kysy kaikkea kerralla vaan askeleittain tarkentamalla.

- *Koosteet ja yhteenvedot*, joiden avulla toisaalta tiivistetään liian suuri määrä tietoa tai toisaalta varmistetaan että järjestelmä on ymmärtänyt käyttäjää oikein.

- *Räätälöimällä* ohjeiden määrä ja palvelun sisältöä käyttäjän kokemuksen ja tarpeiden mukaan.

Tähän asti ei ole ollut valmistajariippumatonta tapaa kuvata ja toteuttaa äänisovelluksia. Tulossa on kuitenkin XML:n sovelluksia, jotka luultavasti muuttavat tilannetta (VoiceXML--).

3.3.2 Tekstiviestit

Tekstiviestit ovat yleinen tapa tehdä palveluita matkapuhelinverkkoihin. Käytettävyyden kannalta ne tosin aiheuttavat seuraavia ongelmia (suluissa olevat numerot viittaavat ongelmaan liittyvään, sivulla 15 esiteltyyn, Nielsenin heuristiikkaan):

- Palvelun ohjaaminen tapahtuu tekstikomennoilla. Käyttäjä joutuu siis hoitamaan itse sovellustason protokollan. (1)

- Yhden tekstiviestin pituus on rajoitettu 160:een merkkiin ja lisäksi kirjoittaminen on hankalaa pienellä näppäimistöllä. Tämän takia voidaan joutua käyttämään lyhenteitä ja muuten tiivistettyä ilmaisua (2). Lisäksi monimutkaiset, paljon tilaa vaativat operaatiot, voivat vaatia useiden viestien lähettämistä edestakaisin, jolloin käyttäjän muistettavat asiat lisääntyvät (3).

- SMS on siirtotienä epäluotettava, sillä matkalla hukkuneista viesteistä ei välttämättä tule virheilmoitusta. Toisaalta palautteen viipymisen syynä voi olla myös viestikeskuksen tilapäinen ruuhkautuminen, joten käyttäjä ei välttämättä saa palautetta reaaliajassa eikä siten tiedä, pitäisikö viesti lähettää uudelleen. (5)

- Käyttöliittymänä on tekstieditori, joka ei tiedä, kirjoittaako käyttäjä oikeita vai virheellisiä komentoja. (9)

- Näytöllä on yleensä vain yksi viesti kerrallaan, joten aiempia viestejä on hankalaa käyttää muistin tukena (3 ja 10). Lisäksi puhelimessa on rajallinen määrä muistia vanhojen viestien tallentamiseen eivätkä ne yleensä tarjoa hakutoimintoa tai vanhojen viestien automaattista poistoa. (10)

Mainittuja käytettävyysongelmia voivat ratkoa sekä päätelaitteiden että palvelujen suunnittelijat. Päätelaitetasolla tekstiviestien käsittelyn pitäisi olla helppoa ja joustavaa samoin kuin tekstin syöttämisen. Tekstiviesteillä käytettävien palvelujen taas pitää olla mahdollisimman "atomisia", toisin sanoen ne eivät saa vaatia pitkiä komentosarjoja tai useiden viestien lähettämistä edestakaisin. Lisäksi palvelun kannattaa käyttää niin paljon kuin mahdollista muita kuin käyttäjän itse syöttämiä tietoja – esimerkiksi käyttäjä voidaan tunnistaa puhelinnumeron perusteella viestiin kirjoitettavan käyttäjätunnuksen sijasta.

3.3.3 WAP-selainsovellukset

WAP on langattomiin verkkoihin ja päätelaitteisiin suunniteltu protokolla-, kuvauskieli- ja rajapintapaketti, joka muistuttaa monessa suhteessa Internetiä ja selainpohjaista WWW:tä.

WAP-protokollat voidaan toteuttaa erilaisten siirtoprotokollien päälle. Esimerkiksi GSM-ympäristössä näitä voivat olla tekstiviestit tai datayhteys. Tästä seuraa kaistanleveyden pienuus, suuremmat viiveet ja huonompi luotettavuus verrattuna Internetiin.

Protokollatasoa enemmän WAP ratkaisee langattomuuden aiheuttamia ongelmia sovellus- ja käyttöliittymätasolla; tähän liittyvät sisällön ja käyttöliittymän kuvauskieli (WML) ja rajapinnat puhelimen toimintojen ohjaamiseen (puhelujen hallinta, kalenteri, puhelinnumeroluettelo).

WML on XML:n sovellus, joka muistuttaa ominaisuuksiltaan HTML:ää. Se sisältää vähemmän mahdollisuuksia tekstin semanttiseen merkitsemiseen, mutta toisaalta taas enemmän piirteitä navigointiin (taulukko 2). WML:n parempi navigointi tarkoittaa

käytännössä sitä, että kieli sisältää toimintoja, jotka muuten tehtäisiin ohjelmoimalla (historian käsittely sekä sisällön automaattiset päivitykset ja siirtymiset paikasta toiseen). Erona näiden kahden välillä on myös se, että WML-dokumentti voidaan jakaa erillisiin kortteihin, joista on kerrallaan näkyvissä yksi. Tämä ominaisuus vähentää näytöllä olevan tiedon määrää, mutta toisaalta ei aiheuta tarvetta muodostaa aina uutta yhteyttä palvelimeen kun näytöllä olevaa tietoa vaihdetaan.

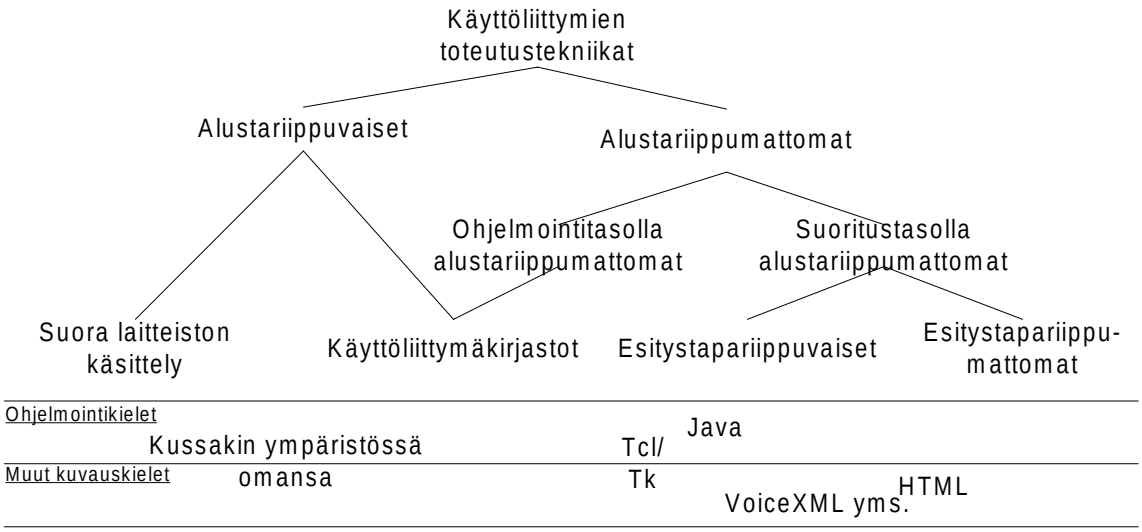
Taulukko 2. HTML:n (W3C 1998c) ja WML:n (WAP Forum 1999) ominaisuuksien vertailu käytössä olevien elementtien määrien perusteella.

	WML 1.1	HTML 4.0
Dokumentin rakenne	6	18
Navigointi	10	2
Lomakkeet	6	10
Listat	0	6
Taulukot	3	9
Teksti (otsikot, kappaleet, korostukset, ...)	9	34
Viittaukset muihin mediatyyppeihin (esim. kuvat)	1	2
Muut	1	

Tekstiviestipalveluihin verrattuna WAP:n käyttö on edistysaskel käyttäjän kannalta: komentojen kirjoittamisen sijasta toiminnot voidaan valita listasta. Käyttötapa on tuttu ainakin vanhoille WWW:n käyttäjille. Ohjeet ja palaute voivat olla runsaampia ja enemmän reaaliaikaisia. Toisaalta palvelujentarjoajien kannalta on epäselvää, mihin tarvitaan uutta WML-kuvauskieltä, joka on kuitenkin melko lähellä HTML-kieltä, mutta ei kuitenkaan yhteensopiva sen kanssa.

3.4 Yhteenveto

Edellisissä luvuissa on kuvattu erilaisia päätelaitteita sekä niihin liittyviä käyttöliittymäratkaisuja. Tässä luvussa esitetään lyhyt yhteenveto aiemmista kappaleista (taulukko 3) ja arvioidaan käyttöliittymien toteutustekniikoiden sopivuutta alustariippumattomiin sovelluksiin; alustariippumattomuus tarkoittaa tässä sekä riippumattomuutta laite- ja ohjelmistoalustasta että riippumattomuutta modaliteetistä eli näyttö- ja syöttölaitteista.



Kuva 7. Käyttöliittymien toteutustekniikoiden luokittelu.

Käyttöliittymien toteutustekniikat voidaan jakaa ylimmällä tasolla sen mukaan (kuva 7), onko tiettyyn käyttöjärjestelmä- ja laiteympäristöön toteutettu sovellus siirrettävissä suoraan muihin ympäristöihin. Alustariippuvaiset tekniikat on jaettu edelleen suoraan laitteistoa käsitteleviin, jolloin sovellus on täysin sitoutunut valittuihin syöttö- ja näyttölaitteisiin, sekä ylemmän tason kirjastoja käyttäviin. Tosin näitä käyttöjärjestelmien ja ohjelmointityökalujen tarjoamia kirjastoja ja rajapintoja on olemassa monentasoisia. Jotkut niistä ovat siirrettävyyden suhteen joustavampia kuin toiset, jotkut tavat voivat sallia esimerkiksi puheohjauksen käyttämisen sovellukselle läpinäkyvästi.

Alustariippumattomat tekniikat on jaettu ohjelmointi- ja suoritustasoisiin. Ensin mainittu tarkoittaa sitä, että ohjelmointi on alustariippumatonta ja siirtäminen tapahtuu kääntämällä ohjelma erikseen kaikkiin niihin ympäristöihin, joita valittu kirjasto tukee, sillä käännettyä sovellusta ei voida enää siirtää. Esimerkiksi puheohjauksen käyttö ilman sen toteuttamista itse sovelluksessa riippuu tässäkin tapauksessa siitä, tukeeko valittu kirjasto tai alla oleva käyttöjärjestelmä sitä.

Suoritustasolla alustariippumattomat käyttöliittymätekniikat adaptoituvat alla olevalle alustalle suorituksen aikana. Kullekin alustalle on olemassa tulkki (Java-terminologiassa virtuaalikone tai HTML-maailmassa selain), joka toimii sovelluksen ja käyttöjärjestelmän välissä. Edelleen on olemassa tekniikoita, jotka ovat ainakin jossain määrin riippuvaisia esitystavasta tai suunnittelusta – esimerkiksi työasemaan suunniteltu graafinen Java-sovellus ei välttämättä ole käyttökelpoinen pienellä näytöllä varustetussa laitteessa. Tällaisiin tilanteisiin sopivatkin paremmin esitystapariippumattomat tekniikat, jotka eivät ole niin tiukkaan sidottuja alustan ominaisuuksiin.

Kuvan 7 luokittelussa siirrettävyys lisääntyy vasemmalta oikealle mentäessä. Luokittelu ei ole diskreetti, vaan myös kategorioiden sisällä on eroja. Esimerkiksi ympäristöriippumaton tekniikka voi olla toista riippumattomampi sen takia, että siihen liittyvä tulkki on toteutettu useammalle alustalle. Myös toteutuksen laatu vaikuttaa asiaan – esimerkiksi huonosti toteutettu HTML-sovellus ei välttämättä ole muita vaihtoehtoja adaptiivisempi. Lisäksi kannattaa muistaa sekin, että pyrittäessä parempaan siirrettävyyteen voidaan joutua tinkimään suorituskyvystä tai joidenkin erikoispiirteiden käytöstä.

Taulukko 3. Päätelaite–sovellustyyppien vertailu.

	Työpöytäsovellukset	WWW	Mukana kuljetettavat	Äänisovellukset	Tekstiviestit	WAP
Vuorovaikutustavat (ks. s. 15)	Kaikki	Navigointi--	Kaikki (rajoituksin)	Komento--, luonnollinen kieli	Komento--	Navigointi--
- Suunta	Kaksisuuntainen	Pyyntö–vastaus	Kaksisuuntainen	Kaksisuuntainen	Kaksisuuntainen	Pyyntö–vastaus
Syöttölaitteet (ks. s. 17)	Ei erityisiä rajoituksia	Ei erityisiä rajoituksia	Pieni näppäimistö, kynä	Puhe, numeronäppäimistö	Pieni näppäimistö	Pieni näppäimistö
Näyttölaitteet (ks. s. 18)	Graafinen näyttö, ääni	Graafinen näyttö, ääniselaimet	Graafinen näyttö	Ääni	Pieni näyttö	Pieni näyttö
Käyttöliittymien kuvauskielet	Lukuisia	HTML, XML:n sovellukset	Lukuisia	Valmistajakohtaiset, VoiceXML	Teksti	WML
Verkko (ks. s. 19)	Kiinteä	Kiinteä	Langaton	Kiinteä / langaton	Langaton	Langaton
- Yhteyden taso	Nopea	Yleensä nopea	Hidas	Välttävä äänenlaatu	Erittäin hidas	Hidas
- Päätelaitteen / käyttäjän liikkuvuus	Huono	Riippuu toteutuksesta (palomuurit)	Hyvä	Ei (kiinteä) / Hyvä (langaton)	Hyvä	Hyvä
Erityisongelmat					Huono käytettävyys	

4. Adaptiivinen käyttöliittymä

Nykysuomen sanakirja (1982) määrittelee adaptiivisuuden sanoilla sopeutuva, mukautuva, sopeutumis- ja mukautumiskykyinen. Tässä luvussa määritellään tarkemmin adaptiivinen käyttöliittymä sekä adaptoivan järjestelmän rakentamiseen liittyviä käsitteitä.

4.1 Määritelmä

Preecen (1994, s. 43–44) mukaan ihmisen ja koneen välisessä vuorovaikutuksessa on mukana neljä komponenttia: ihmiset, työ, ympäristö ja teknologia. Komponentit ovat vuorovaikutuksessa toistensa kanssa siten, että jos yksi muuttuu, täytyy muidenkin muuttua (adaptoitua). Esimerkiksi teknologinen muutos muuttaa ihmisten tapaa tehdä työnsä, joka taas vaikuttaa fyysiseen, sosiaaliseen ja organisationaaliseen ympäristöön.

Benyonin (1998) mukaan järjestelmä voidaan adaptoida ympäristöönsä hyvällä suunnittelulla, jossa voidaan ottaa huomioon käyttäjien ajan myötä muuttuvat vaatimukset. Tästä ovat esimerkkinä useat pelit, joiden vaikeustasoa voidaan nostaa pelaajan taitojen lisääntyessä. Hyvä suunnittelu pysyy kuitenkin hyvänä vain niin kauan, kuin käyttö pysyy ennalta määriteltyjen vaatimuksien rajoissa. Benyonin ratkaisu tähän ongelmaan on dynaamisen adaptoinnin käyttö eli itseään adaptoiva tai muuten adaptoitava järjestelmä, jonka tila tai käyttäminen muuttuu tarkoituksenmukaisesti saamiensa signaalien mukaan.

Kuva 8. Ihmisen ja koneen vuorovaikutukseen vaikuttavat tekijät (Preece 1994).

Adaptointi voi tapahtua minkä tahansa käyttöliittymään vaikuttavan tekijän suhteen (kuva 8). Alla on listattu joitakin näkökulmia adaptiivisuuteen (kukin voidaan käsittää staattisena, suunnittelun yhteydessä tapahtuvana tai dynaamisena, käytön aikana tapahtuvana):

- Käyttäjän kykyihin ja ominaisuuksiin sopeutuminen. Käyttöliittymä voi adaptoitua esimerkiksi käyttäjän kulttuuritaustan, kieliryhmän, käyttökokemuksen tai fyysisten ominaisuuksien mukaiseksi.

- Käyttäjän työtehtäviin, kiinnostuksiin tai tarpeisiin sopeutuminen.

- Käyttötilanteen tarpeisiin tai kontekstiin sopeutuminen.

- Organisaation tarpeisiin tai tavoitteisiin sopeutuminen (vrt. BPR, business process re-engineering).

- Teknisiin laitteisto- ja ohjelmistotekijöihin sopeutuminen.

Tässä tutkielmassa adaptoinnilla tarkoitetaan laitteisto- ja ohjelmistoteknisiin seikkoihin sopeutumista. Tällöin adaptointiin vaikuttavia tekijöitä ovat ainakin seuraavat:

1. Päätelaitteen fyysiset ominaisuudet, joista tärkeimpiä ovat tuetut syöttö- ja näyttölaitteet sekä niiden ominaisuudet (esimerkiksi koko). Myös prosessorin teho tai muistin määrä voi rajoittaa käyttöliittymää.

2. Päätelaitteen tukemat vuorovaikutustavat ja käyttöliittymät. Tähän vaikuttavat edellisessä kohdassa mainittujen fyysisten ominaisuuksien lisäksi myös suunnittelupäätökset, joita laitteiden tai niiden sisältämien ohjelmien suunnittelija on tehnyt.

3. Hajautusratkaisu ja siinä käytettävät verkot ja protokollat. Hajautusratkaisuun vaikuttavat myös osaltaan laitteen fyysiset ominaisuudet eli se, mikä osa järjestelmää on mahdollista ja järkevää toteuttaa missäkin. Esimerkiksi joissain tapauksissa voi olla tarkoituksenmukaista hajauttaa käyttöliittymän esitys, sovelluslogiikka ja tietovarasto eri paikkoihin.

Adaptointi voi tapahtua ylös- tai alaspäin (Lie & Saarela, 1999). Alaspäin tapahtuva adaptointi tarkoittaa karsintaa, koska ominaisuuksiltaan rajoittuneempi päätelaite ei tue kaikkia alkuperäisen esityksen piirteitä. Ylöspäin tapahtuva adaptointi on tälle vastakkainen prosessi, jossa esitystä rikastetaan ennalta määrättyjen sääntöjen avulla.

4.2 Adaptiivisuuden toteuttaminen

Tässä luvussa määritellään komponentit, joiden avulla adaptiivisuus voidaan toteuttaa dynaamisesti käytön aikana: adaptoiva järjestelmä sekä sen vaatima kuvauskieli (abstrakti käyttöliittymämalli), jonka avulla voidaan määritellä adaptoitava käyttöliittymä.

4.2.1 Adaptoiva järjestelmä

Kuten johdannossa mainittiin on eri päätelaitteiden suhteen adaptiivisella järjestelmällä yhtymäkohtia vammaisille toteutettavien käyttöliittymien kanssa. Vammaisten huomioonottamisessa järjestelmien suunnittelussa ja toteutuksessa on neljä tapaa (käytettävyys vastaa tässä englannin kielen sanaa accessibility) (Vanderheiden & Vanderheiden 1992):

1. *Suora käytettävyys.* Järjestelmää suunniteltaessa on otettu mukaan tekijöitä, jotka mahdollistavat erilaiset käyttötavat, kuten näppäimistön tai peliohjaimen käytön hiiren sijasta.

2. *Järjestelmään liitettävät ylimääräiset osat.* Jotkut käytettävyyttä lisäävät tekijät voivat olla toisensa poissulkevia, tavallista käyttöä haittaavia tai niiden integroiminen voi olla teknisesti tai taloudellisesti muuten liian hankalaa, joten järjestelmään lisätään mahdollisuus komponenttien lisäämiseen tai korvaamiseen.

3. *Kolmansien osapuolien ratkaisut.* Esimerkiksi käyttöjärjestelmässä voi olla ominaisuuksia, kuten vaihtoehtoisten syöttölaitteiden käyttö, joiden ansiosta kaikissa toteutettavissa järjestelmissä ei tarvitse ottaa käytettävyyttä huomioon.

4. *Järjestelmän tekeminen konfiguroitavaksi tai laajennettavaksi.*

Tutkielman kohteena olevaa Princess-järjestelmää voidaan pitää edellä mainittuna kolmannen osapuolen ratkaisuna, sillä se pyrkii abstrahoimaan eri päätelaitteet palveluntarjoajalle. Ei kuitenkaan ole aivan realistista olettaa, että päätelaite pystytään täysin peittämään palvelujen tarjoajilta, joten järjestelmää suunniteltaessa on otettava huomioon myös ensimmäinen vaihtoehto.

4.2.2 Abstrakti käyttöliittymämalli

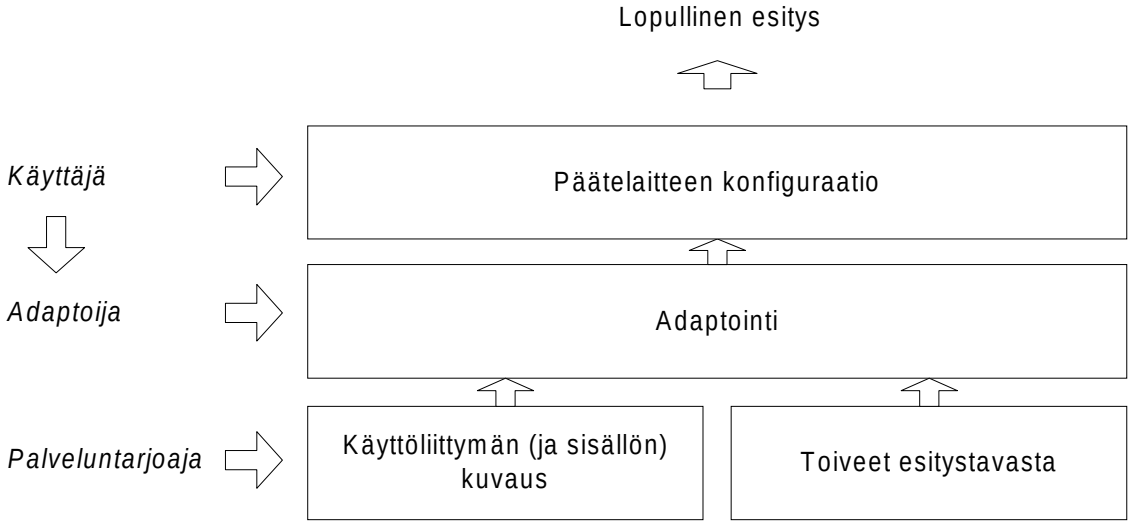
Jotta päätelaitteen abstrahointi palveluntarjoajalle olisi mahdollista, tarvitaan päätelaiteriippumaton kuvaustapa, jolla palvelujen käyttöliittymä ja sisältö voidaan kuvata. Princess-projektissa tätä kuvaustapaa kutsutaan termeillä abstrakti käyttöliittymämalli tai presentaatiomalli (presentation model). Molemmilla termeillä tarkoitetaan käytännössä samaa asiaa, mutta ensin mainitulla korostetaan käyttöliittymänäkökulmaa ja jälkimmäisellä yleisemmin sisällön esittämistä.

Kuvaustapaa suunniteltaessa voidaan soveltaa Vanderheidenin (1997) esittämiä strategioita: voidaan valita modaliteetistä riippumaton tai modaalisesti redundanttinen esitys. Ensin mainittu perustuu siihen, että määritellään esitystapariippumaton kuvaus, joka on adaptoitavissa eri ympäristöihin. Toinen vaihtoehto on tehdä redundanttinen kuvaus, jossa samaan esitykseen on sisällytetty useita vaihtoehtoisia tapoja esittää sama asia.

Joitakin kuvaustapoja on käsitelty aiemmissa luvuissa (kuva 7). Valintaa tehdessä täytyy kiinnittää huomiota esitystapariippumattomuuden lisäksi myös käyttökohteeseen: esimerkiksi HTML-kieli on suhteellisen yleiskäyttöinen, kun taas XML:n avulla voidaan määritellä kuvauskieli tarvittaessa hyvinkin kapealle sovellusalueelle.

Kuvauskielen valintaan liittyy myös kysymys siitä, kenen annetaan vaikuttaa esitystapaan ja kuinka paljon (kuva 9). Adaptoiva järjestelmä ja päätelaite luonnollisesti päättävät esitystavan, mutta myös käyttäjä voi haluta tehdä omia asetuksiaan, ja sisällön tuot-

tajalle voi olla tärkeää se, miten hänen tuottamansa sisältö esitetään. Hyvä esimerkki tavasta, jolla kaikki edellä mainitut voivat vaikuttaa esitystapaan ovat CSS-tyylitiedostot (luku 3.1.2).



Kuva 9. Käyttöliittymän ja sisällön lopullisen esitystavan määräytyminen.

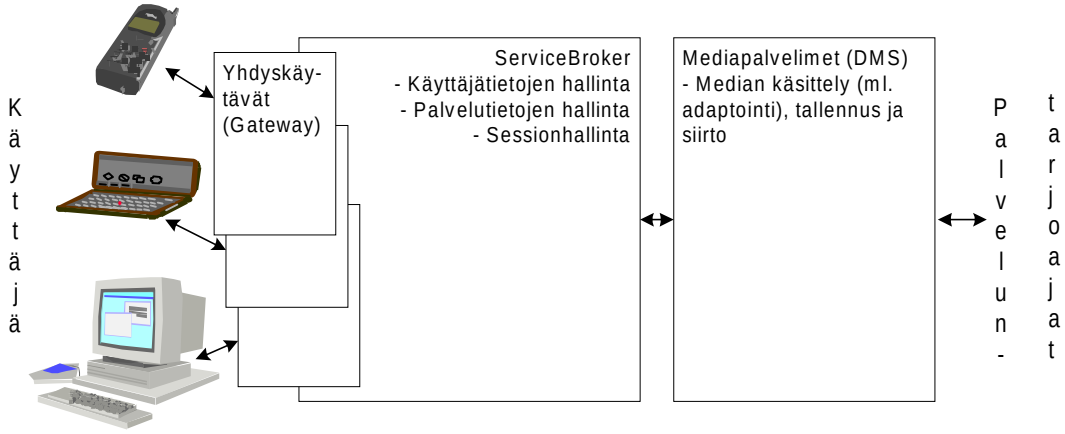
Käyttäjän kannalta luonnollinen tapa vaikuttaa esitystapaan on päätelaitteen valinta ja konfiguraatio: esimerkiksi WWW-selaimiin sisältyy yleensä useita ominaisuuksia, jotka voi säätää mieleisekseen. Lisäksi myös adaptoiva järjestelmä voidaan rakentaa siten, että se tunnistaa käyttäjänsä ja ottaa tämän huomioon.

Palveluntarjoajan kannalta ulkoasun määrääminen on monimutkaisempi ongelma: toisaalta adaptoivan järjestelmän idea on abstrahoida päätelaite, mutta toisaalta voi olla tarpeen optimoida palvelun ulkoasua tai toimintaa myös päätelaitekohtaisesti. Pitkälle vietynä tämän kaltainen optimointi asettaakin koko adaptiivisen järjestelmän hyödyllisyyden kyseenalaiseksi.

5. Adaptiivisen käyttöliittymän toteutus

Tässä luvussa pyritään vastaamaan toteutukseen liittyvään kolmanteen alatutkimusongelmaan (s. 11) Princess-projektissa tehdyn työn perusteella. Toteutuksen yhteydessä käytetyt käsitteet adaptiivinen käyttöliittymä ja abstrakti käyttöliittymämalli on määritelty luvussa 4.

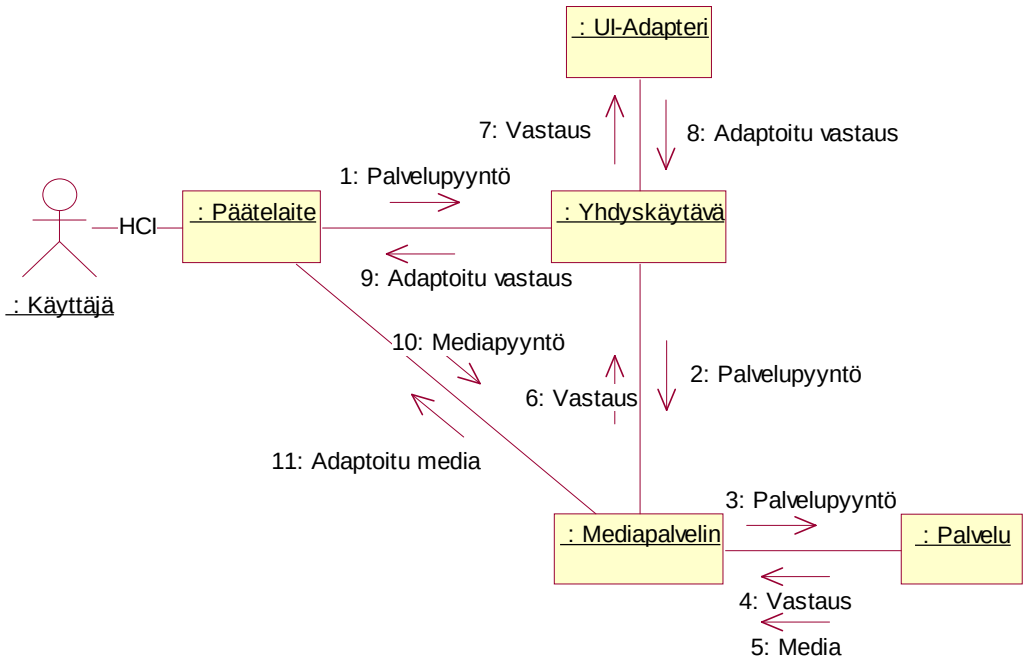
5.1 Princess-järjestelmän arkkitehtuuri



Kuva 10. Princess arkkitehtuuri.

Princess-arkkitehtuurin osia (kuva 10) ovat palveluntarjoajan rajapinta, digitaalinen mediapalvelin (DMS), ServiceBroker ja päätelaitekohtaiset yhdyskäytävät. Järjestelmän keskipiste on ServiceBroker, jonka tehtävänä on yhdistää käyttäjä ja palvelu toisiinsa sekä mahdollistaa käyttäjän liikkuvuus. DMS taas adaptoi median, kuten äänen, kuvan ja videon, päätelaitteen ja verkon kykyjä vastaavaksi sekä sisältää palveluntarjoajan työkaluja esimerkiksi videon indeksointiin. Yhdyskäytävät hoitavat käyttöliittymän esityksen ja siihen liittyvän vuorovaikutuksen päätelaitteen tukemalla tavalla. Toisin sanottuna ne vastaavat abstraktin käyttöliittymämallin muuntamisesta päätelaitteen tukemaan muotoon sekä tarvittavasta siirtoprotokollasta.

Kuva 11 esittelee järjestelmän toimintaa tilanteessa, jolloin järjestelmä on käynnistetty ja käyttäjä on muodostanut yhteyden aiemmin. Kuvassa ovat järjestelmän osat ServiceBrokeria lukuun ottamatta, koska sen osuus rajoittuu lähinnä käyttäjän sisäänkirjautumiseen ja palvelun valintaan.



Kuva 11. Princess-järjestelmän toiminta UML-yhteistyödiagrammina.

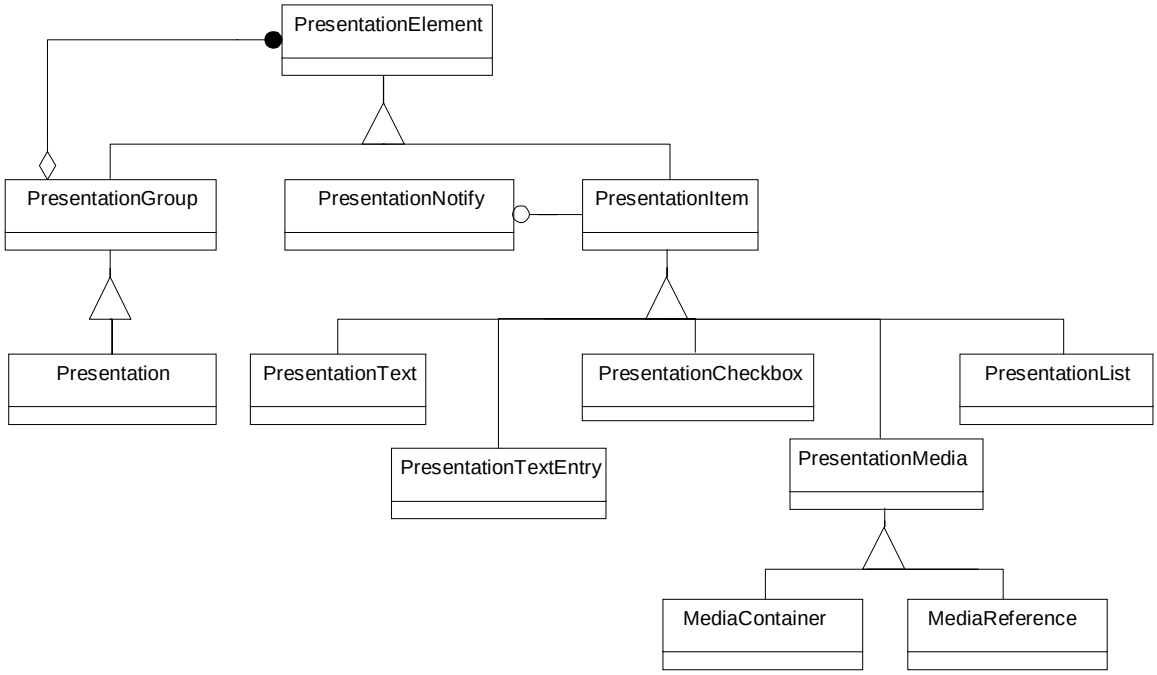
Järjestelmän suunnittelun yhtenä päämääränä on ollut hajautus, jolla pyritään paitsi jakamaan paljon suoritinkapasiteettia vaativia toimintoja eri laskentayksiköille myös optimoimaan verkon käyttöä sallimalla järjestelmän komponenttien sijoittelu eri paikkoihin. Järjestelmään voi kuulua myös useita ServiceBrokereita ja mediapalvelimia, jotka eivät välttämättä ole fyysisesti tai loogisesti samassa paikassa. Kannattaa myös huomata, että järjestelmän tehokkuuden optimoimiseksi on päädytty ratkaisuun, jossa päätelaite hakee median suoraan mediapalvelimesta eikä yhdyskäytävän kautta (kuva 11).

5.2 Abstraktin käyttöliittymämallin valinta

Abstrakti käyttöliittymämalli on Princessissä ensisijaisesti järjestelmän sisäinen kuvaustapa, joka adaptoidaan yhdyskäytävässä päätelaitekohtaiseksi esitykseksi. Tosin myös palveluntarjoajat voivat käyttää sitä palvelujen kuvaamiseen, vaikka se ei olekaan pakollista. Hyvä esimerkki tilanteesta, jossa palveluntarjoaja ei mallia käytä on videopalvelu: järjestelmään tuotu sisältö voi olla TV-lähetys, joka indeksoidaan Princess-järjestelmän tarjoamilla työkaluilla ja tallennetaan tietokantaan katsottavaksi myöhemmin järjestelmässä olevan sisäisen palvelun avulla.

Ensimmäinen Princess-projektissa toteutettu käyttöliittymämalli oli Java-pohjainen oliomalli, jolla pystyi toteuttamaan HTML-lomaketta vastaavan toiminnallisuuden. Tällöin tarkoituksena oli käyttöliittymän ja median (palvelun sisällön) erottaminen toisistaan eri

ikkunoihin tai kokonaan eri päätelaitteisiin. Pian kuitenkin huomattiin, että toiminnallisuuden ja sisällön erottaminen toisistaan ei ole käytännössä järkevää: hypertekstijärjestelmässä, joka Princesskin pitkälti on, voidaan esimerkiksi linkit lukea kumpaankin kategoriaan. Ongelman ensimmäinen ratkaisu oli lisätä malliin PresentationMedia-luokat, jolla samassa esityksessä pystyi siirtämään mediaa joko upottamalla tai viittaamalla (kuva 12). Ensin mainittu tapa kuljettaa päätelaitteen tunteman median, kuten HTML-koodin, oliomallin mukana, jälkimmäinen siirtää viittauksen paikkaan, jossa esimerkiksi video tai kuva sijaitsee.



Kuva 12. Ensimmäisen vaiheen oliomalli.

Käytännössä tämäkin sisällön ja käyttöliittymän erottelu kuvattavaksi usealla eri tavalla johti siihen, että esimerkiksi henkilötietojen hakua demonstroineessa palvelussa käyttäjä sai ensimmäisenä listan organisaation työntekijöistä, jonka alla oli käyttöliittymäosa, johon hän pystyi kirjoittamaan haluamansa henkilön nimen tarkempien tietojen saamista varten. Yksinkertaisinta olisi ollut valita henkilö suoraan listasta, mitä toteutus ei kuitenkaan sallinut.

Tässä vaiheessa olisi ollut mahdollista jatkaa valitulla linjalla kehittämällä oliomallia niin, että sillä olisi voinut kuvata kaiken tekstimuotoisen sisällön, viittaukset binäärimuotoisiin mediaolioihin, käyttöliittymäkomponentit sekä vielä vaikuttaa monipuolisesti esityksen ulkoasuun. Toisaalta vaihtoehtona oli sellaisen valmiin ratkaisun hakeminen, joka täyttäisi seuraavat vaatimukset:

- Kyky kuvata käyttöliittymä ja sisältö esitystapariippumattomasti. Palvelutyyppejä ei rajata mitenkään, joten ratkaisun pitää olla mahdollisimman yleiskäyttöinen.

- Valmis, standardipohjainen ratkaisu, jota voidaan tarvittaessa itse laajentaa.

- Valmiiden työkalujen ja rajapintojen olemassaolo, joilla voidaan luoda, muokata, tallentaa ja muuntaa esityksiä.

Valmiita ratkaisuja, joita voidaan käyttää sisällön ja käyttöliittymän kuvaamiseen, olivat ainakin MHEG ja eräät rakenteellisten kuvauskielten, SGML:n ja XML:n, sovellukset.

MHEG-tiedonsiirtoformaatin tarkoitus on määritellä laajamittaisten telemaattisten multimediasovellusten lopullinen ulkoasu. Se on tarkoitettu käytettäväksi kiinteissä verkoissa ja yleislähetysverkoissa (broadcast), kuten digitaalitelevisioissa (Price 1993). Siksi se sopii huonosti adaptoitavaksi langattomiin verkkoihin ja käyttöliittymäominaisuuksiltaan rajoittuneisiin päätelaitteisiin. Lisäksi työkalutuki on heikko. Näistä syistä MHEG hylättiin varsin nopeasti.

SGML ja XML ovat rakenteellisen tiedon kuvaamiseen soveltuvia kuvaustapoja. Tunnetuin SGML:n sovellus on HTML, jota käytetään yleisesti verkkopalvelujen käyttöliittymien ja sisällön kuvauksessa Internetissä. XML toisaalta on SGML:n osajoukko, mikä helpottaa sovelluksien ja työkalujen kehitystä. Tästä syystä se valittiin tarkempien tutkimusten kohteeksi, jolloin päädyttiin seuraaviin tuloksiin (tulokset raportoitu tarkemmin sisäiseen käyttöön tehdyssä raportissa Korva & Metso (1999); ks. myös Liite B):

- Vaikka XML ei olekaan vielä kovin vanha ilmiö, on sille lukuisia sovelluksia eri aloilta, kuten metatiedon esittäminen (RDF – resource description framework), elektroninen kaupankäynti ja tiedonsiirto sekä sisällön ja käyttöliittymän kuvaus (SMIL – synchronized multimedia integration language, WML – wireless markup language ja XHTML – extensible hypertext markup language).

- XML-pohjaiset ratkaisut ovat hyvin laajennettavia: on mahdollista luoda omia sovelluksia, yhdistää olemassa olevia tai määritellä niihin lisäyksiä.

- XML on avoin ratkaisu ja se ei ole riippuvainen mistään laite-, käyttöjärjestelmä- tai ohjelmointiympäristöstä.

- Saatavilla on paljon työkaluja XML-dokumenttien käsittelemiseen, jotka ovat myös usein ilmaisia ja Java-pohjaisia.

Merkittävin huono puoli XML:ssä on keskeneräisyys: työkalut ovat vielä kaikki kehitysvaiheessa sekä moni liitännäisstandardi on hyväksymättä.

HTML:n perillinen, XHTML, vastasi hyvin Princessin vaatimuksia, sillä muista XML-pohjaisista vaihtoehdoista SMIL on suunniteltu multimediaesityksien tekemiseen (siinä on samoja piirteitä ja ongelmia kuin MHEG:ssä) ja WML on suunniteltu vain pieniä laitteita varten. Toisaalta olisi ollut myös mahdollista suunnitella kokonaan oma dokumenttityyppi, mutta tätä ei koettu mielekkääksi.

XHTML 1.0 sisältää täsmälleen samat elementit kuin HTML 4.0: linkit, listat, lomakkeet, taulukot jne. Ainut merkittävä ero on se, että dokumenttien täytyy noudattaa entistä tiukempaa XML-syntaksia. (W3C 1999b.)

Esitystavan määrittämiseen on HTML:n ja XHTML:n osalta kaksi mahdollisuutta: tyylitiedostot, lähinnä siis CSS, sekä niin sanottu transitional-dokumenttityyppi, joka sisältää elementtejä, joilla ulkoasu voidaan määritellä HTML-koodissa (esimerkiksi `<font>`, `<center>` jne.). Princess-projektissa pyritään käyttämään ensin mainittua muun muassa seuraavista syistä.

- CSS:n avulla voi määritellä useita sekä visuaalisuuteen että ääneen perustuvia esityksiä (luku 3.1.2). Transitional-dokumenttityypillä voi määrittää vain yhden visuaalisen esityksen.

- CSS:n avulla käyttäjä voi vaikuttaa itse monipuolisemmin palvelujen ulkoasuun, joko omien mieltymyksiensä tai fyysisten rajoituksiensa, kuten huonon näkökyvyn, mukaisesti (W3C 1998a).

- Transitional-ominaisuuksia ei todennäköisesti sisälly tuleviin XHTML-versioihin (W3C 1999d).

Projektissa suunniteltiin myös joitakin laajennoksia XHTML:ään (Liite B). Merkittävin laajennos on sisällön semanttinen priorisointi: princess:priority-attribuutin avulla voidaan valita dokumentista tärkeimmät osat, jotka on oleellista näyttää jokaisella päätelaitteella, jos koko esitystä ei voida käyttää (kuva 13). Muita laajennoksia tehtiin mm. laskutusta varten.

```
<?xml version="1.0"?>
<html xmlns="http://www.w3.org/TR/xhtml1"
      xmlns:princess="urn:tekes.fi:princess" xml:lang="fi">
  <head>
    <!-- JKo 28.6.1999 10:53 -->
    <title>Dokumentin otsikko</title>
  </head>
  <body>
    <p princess:priority="-1"><img src="logo.gif" alt="logo"/></p>

    <h1>Otsikko</h1>

    <object data="TheEarth.mpeg" type="application/mpeg">
      <object data="TheEarth.gif" type="image/gif">
        Maapallo avaruudesta katsottuna.
      </object>
    </object>

    <p>
       <span princess:priority="1">Johdantolause.</span> Tarkemmin...
    </p>

    <form action="princess://palvelu/dokumentti">
      <p><label>Nimi <input type="text" name="nimi"/></label></p>
    </form>
    <hr/>
  </body>
</html>
```

Kuva 13. Esimerkki XHTML-dokumentista Princess-laajennoksista.

XHTML:n käyttöönoton yhteydessä tarvittiin myös uusi tapa nimetä järjestelmässä olevat resurssit eli palvelut ja dokumentit. Ratkaisuksi kehitettiin oma URL-tyyppi (taulukko 4) RFC 2396:n (1998) pohjalta.

Taulukko 4. Princess URL eli järjestelmän sisäisen resurssin nimi.

princess://<palvelu>/<alipalvelu>/<dokumentti>?<parametri>=<arvo>#<ankkuri> esim. princess://merisaa/tankar/?paiva=27101999#nakyvyys	
<palvelu>	Palvelun nimi. Palvelun yhteydessä on mahdollista määritellä myös palvelin (DMS), joka tarjoaa palvelun.
<alipalvelu>, <dokumentti>	Hierarkkinen osa, joka määrittää halutun alipalvelun ja dokumentin.
<parametri>, <arvo>	Joukko parametri-arvo -pareja. Käytetään samoin kuin HTTP:nkin yhteydessä.
<ankkuri>	Viittaus dokumentin osaan.

5.3 Käyttöliittymien dynaaminen generointi päätelaitteille

Tässä kappaleessa käsitellään adaptoinnin toteutusta käytännössä eli sitä, miten aiemmin kuvattu abstrakti käyttöliittymämalli saadaan "eloon" eri päätelaitteissa. Princess-arkkitehtuurissa tämä tarkoittaa yhdyskäytävien toteutusta: sitä miten abstrakti käyttöliittymämalli saadaan adaptoitua päätelaitteen tukemaan muotoon, miten käyttöliittymä saadaan esitettyä käyttäjän päätelaitteessa ja miten interaktio käyttäjän ja Princess-järjestelmän välillä tapahtuu.

Adaptoinnin ensimmäinen edellytys on päätelaitteen tunnistaminen. Karkea tunnistaminen tapahtuu yhteydenottotavan mukaan. Näitä tapoja ovat esimerkiksi Java-sovelluksen käynnistäminen, HTTP-yhteyden ottaminen (selaimet) ja tekstiviestin saapuminen. Tarkempi tunnistaminen riippuu täysin yhteydenottotavasta: Java-toteutuksessa voidaan käyttää kielen tarjoamia rajapintoja. Eri selaimet taas voidaan tunnistaa melko hyvin HTTP-pyynnössä olevien tietojen avulla, minkä lisäksi sivulle upotettujen JavaScript-ohjelmien avulla on mahdollista saada joistakin selaimista tarkempia tietoja esimerkiksi näytön resoluutiosta ja värien määrästä.

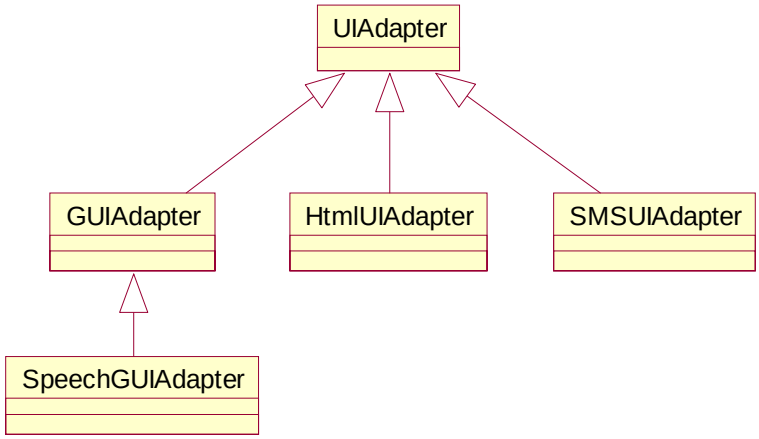
Tässä luvussa käydään läpi ensimmäisen ja toisen vaiheen toteutuksen ja arkkitehtuurin eroja (vaiheet on määritelty luvussa 5.2). Sen jälkeen esitellään toteutus tarkemmin päätelaitetyypeittäin. Yhteenvedossa arvioidaan toteutuksia taulukossa 5 esitellyn kehikon mukaisesti. Luvussa 5.4 esitellään toteutuksessa käytettyjä Java- ja XML-teknologioita ja -työkaluja.

Taulukko 5. Käyttöliittymän adaptoinnin tasot (vrt. Taxonomy of multimedia adaptation, Metso et al. 1998).

0.	ei adaptointia	Päätelaite tukee alkuperäistä esitystapaa
1.	konvertointi	Vastaavat käyttöliittymäelementit ja syöttölaitteet löytyvät, ainoastaan kuvauskieli on eri. Käyttö voi myös tapahtua eri syöttölaitteella, mutta käyttöjärjestelmä kätkee tällaiset muutokset.
2.	skaalaus	Eri käyttöliittymäelementtien välillä ei löydy suoraa vastaavuutta, mutta vastaava toiminto voidaan suorittaa muuten. Esimerkiksi valintalista voidaan muuntaa joukoksi kyllä/ei -kysymyksiä.
3.	optimointi	Muutetaan käyttöprosessia ja/tai interaktiotapaa. Järjestelmä voi esimerkiksi tehdä toimenpiteitä käyttäjän puolesta.

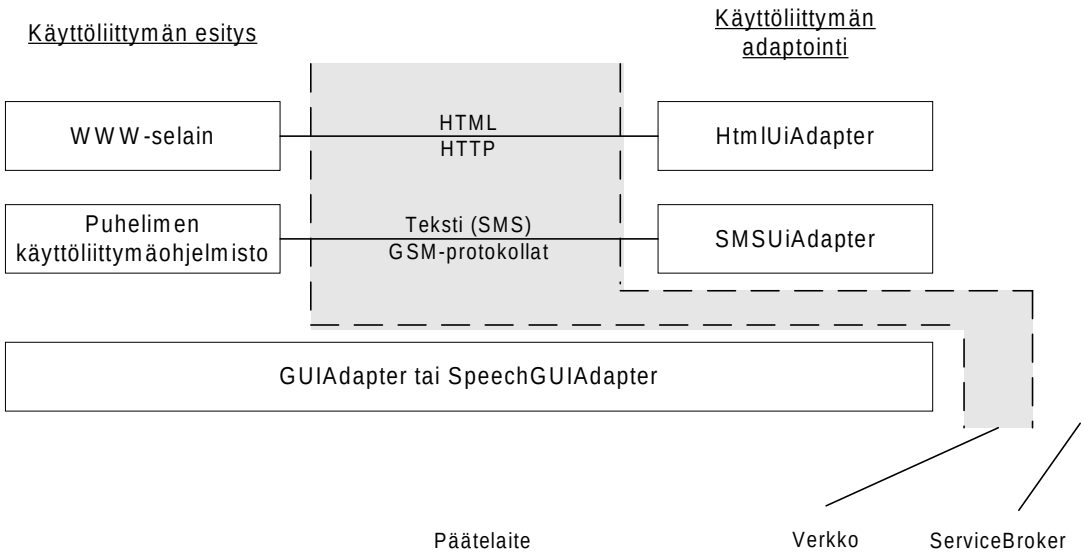
Ensimmäinen vaihe

Projektin ensimmäisessä vaiheessa kuvaustapojen – oliomallin ja sisällönkuvaukseen käytetyn HTML:n – sekoittaminen keskenään aiheutti turhaa työtä ja mutkisti toteutusta, sillä kaikissa järjestelmän osissa piti pystyä käsittelemään molempia (luku 5.2). Käytännössä tämä näkyi esimerkiksi adaptoinnin varsinaisesti suorittavien UIAdapterien (kuva 14) toteutuksessa, joiden täytyi paitsi osata generoida päätelaitteelle sopiva esitys oliomallista myös yhdistää siihen mediaesitykset.



Kuva 14. Princess UIAdapterit ensimmäisessä vaiheessa.

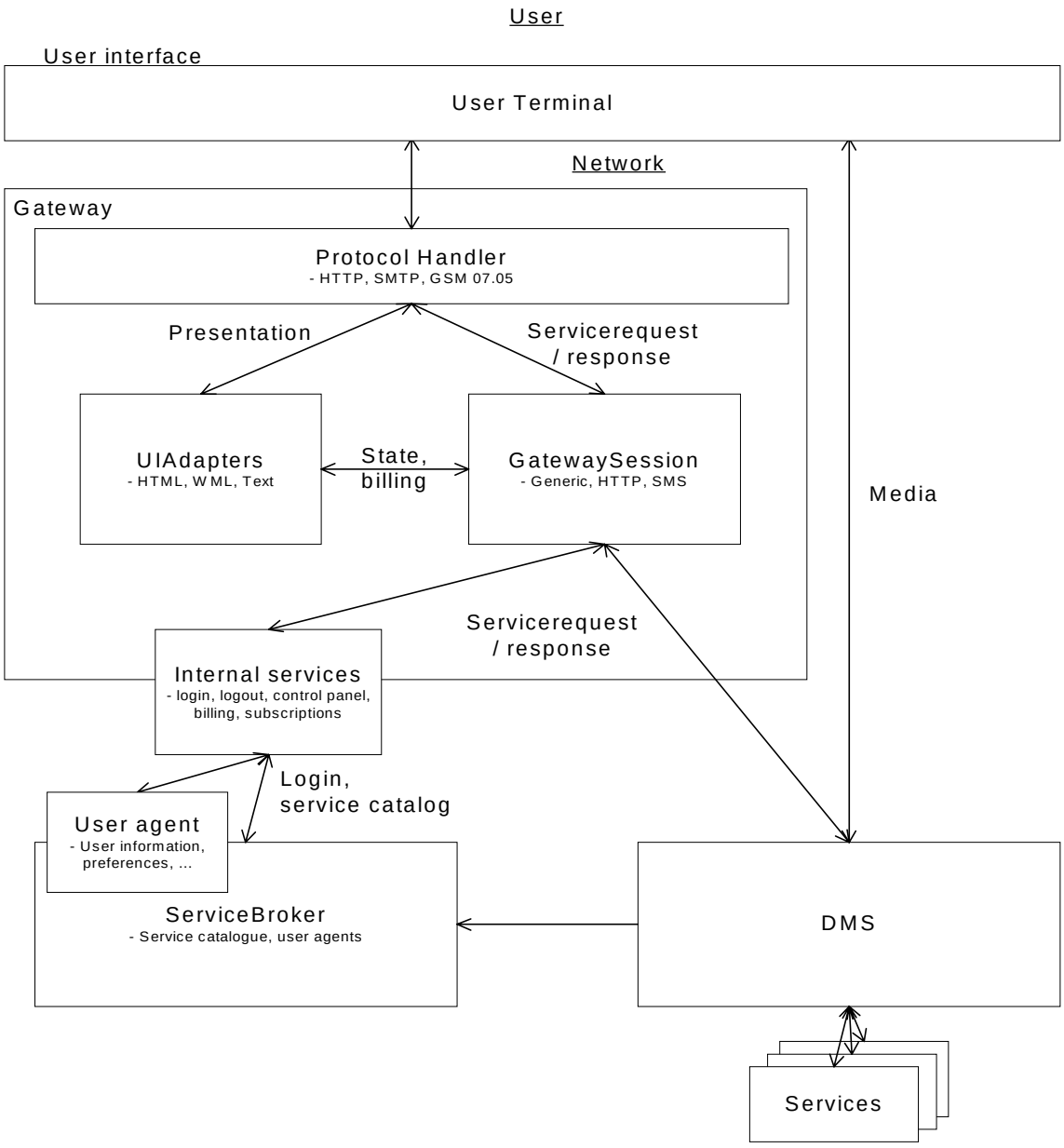
Ensimmäisessä vaiheessa tuettuja käyttötapoja olivat Java-sovellus sekä tavallisena graafisena käyttöliittymänä että puheohjattuna, WWW-selaimet ja tekstiviestit. Näistä Java-sovellus eroaa muista (kuva 15), sillä siinä käyttöliittymän hallinta tapahtuu kokonaan päätelaitteessa, kun se muissa on hajautettu osittain palvelimeen.



Kuva 15. UIAdapter-komponenttien arkkitehtuuri.

Toinen vaihe

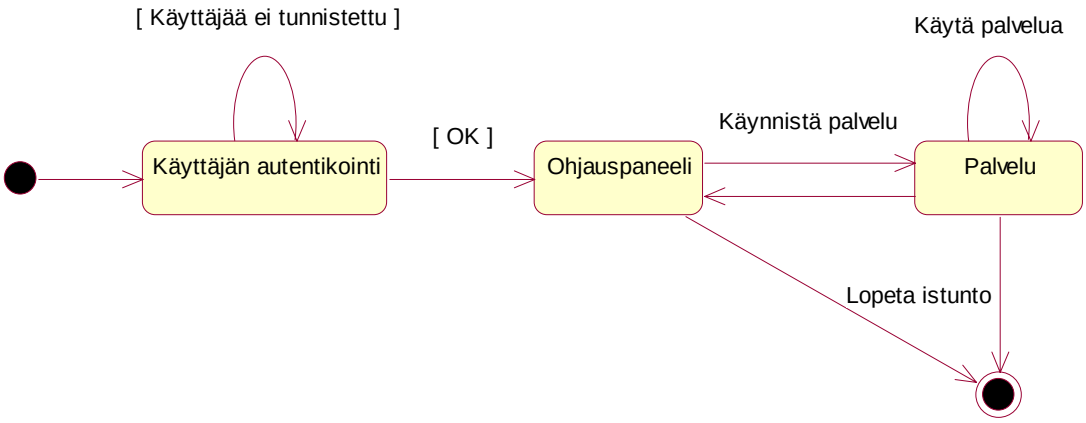
Toisessa vaiheessa otettiin käyttöön uusi XHTML-pohjainen kuvauskieli ja siihen liittyvät työkalut. Lisäksi ohjelmistoarkkitehtuuria muutettiin joustavammaksi paremman modularisoinnin avulla. Ensimmäisenä toteutettiin WWW-, tekstiviesti- ja sähköpostiyhdyskäytävät. Myöhemmin toteutetaan WAP- ja puheliittymät.



Kuva 16. Toisen vaiheen arkkitehtuuri (tutkielmaan liittyvät osat kuvattu muita yksityiskohtaisemmin).

Käyttöliittymätoteutuksen kannalta arkkitehtuurissa tärkeimpiä komponentteja (kuva 16) ovat yhdyskäytävä (gateway), käyttöliittymän adaptoija (UIAdapter), sessio (GatewaySession) ja palvelut (internal services, services). Samat toiminnot sisäisiä palveluja lukuun ottamatta löytyivät kyllä aikaisemmastakin versiosta, mutta nyt ne on erotettu selkeämmin toisistaan. Tällä saavutetaan se etu, että lisättäessä tuki uudelle päätelaitteelle kirjoitetaan vain tarvittavat komponentit uudelleen. Esimerkiksi WAP- ja WWW-selaimet käyttävät järjestelmän kannalta samaa HTTP-protokollaa ja sessionhallintaa, joten ainut uudelleen toteutettava osa on adaptoija.

Sisäiset palvelut ovat geneerisiä tapoja hoitaa järjestelmään sisään- ja uloskirjautuminen, palvelun valinta, käyttäjätietojen hallinta jne. (kuva 17). Kaikkiin vuorovaikutustapoihin ja päätelaitteisiin geneerinen tapa ei kuitenkaan sovi, joten sisäiset palvelut voidaan tarvittaessa ohittaa: esimerkiksi tekstiviestiyhdyskäytävässä sisäänkirjautuminen ja palvelun valinta kannattaa hoitaa suoraviivaisemmin kuin selainyhdyskäytävässä.



Kuva 17. Geneerinen istunto Princess-järjestelmässä.

5.3.1 WWW-selaimet

WWW-selainten integrointi Princess-järjestelmään on helppoa, koska järjestelmän sisäinen kuvauskieli ja interaktiomalli ovat lähellä selainten käyttämiä (Kuva 18). Suurin ongelma selaimissa on yhden yleisesti hyväksytyn ja kunnolla toteutetun standardin puute: Yleisimmissä selaimissa on kussakin omia, muiden kanssa yhteensopimattomia laajennoksia. Lisäksi yhteisesti sovittuja W3C-suosituksia ei noudateta täysin. Tämä lisää toteutuksen monimutkaisuutta, jos halutaan varmistaa toimivuus mahdollisimman suuressa joukossa selaimia.

Kuva 18. Kuvaruutukaappaus videovalvontapalvelusta WWW-selaimessa.

5.3.2 Tekstiviestit

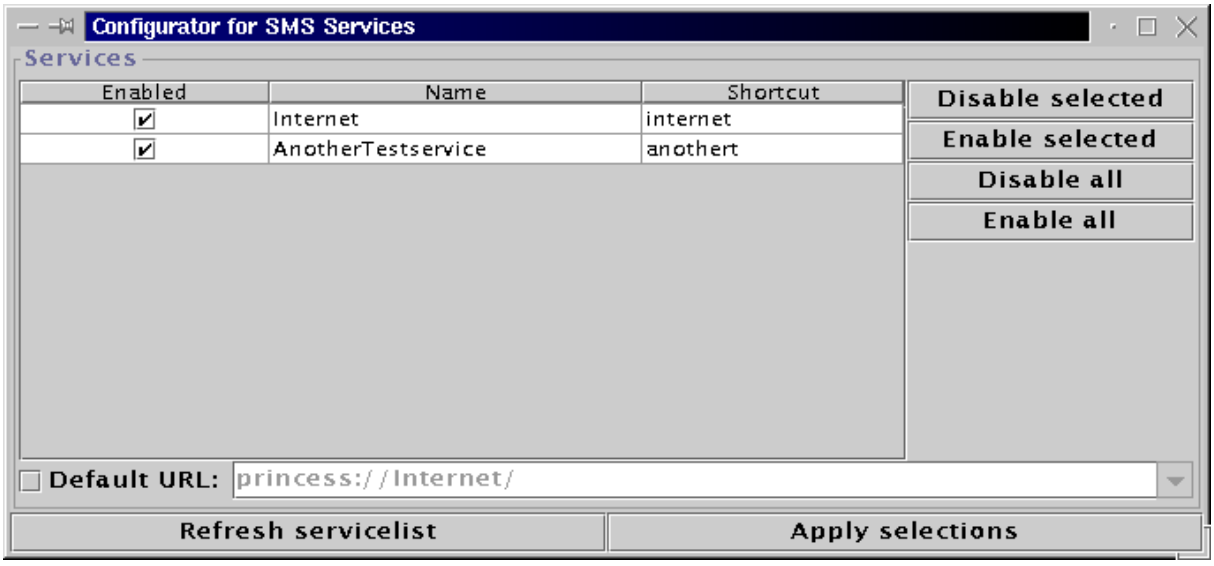
Käytettävyysongelmiensa takia tekstiviestiyhdyskäytävä sopii parhaiten hälytysten ja ilmoitusten välittämiseen palvelusta käyttäjälle (luku 3.3.2). Toisaalta myös interaktiivisen käytön pitää olla mahdollista ainakin niin kauan kun kehittyneemmät tavat, kuten WAP, eivät ole laajasti käytössä.

Tekstiviestien käytettävyyden optimointi interaktiivisessa käytössä vaatii sekä käyttöä helpottavia ominaisuuksia Princess-alustalta että huomiointia palvelun suunnittelussa ja toteutuksessa. Alustan tarjoamia ominaisuuksia ovat helppo sisään- ja uloskirjautuminen, oikopolut, avusteet, järkevät komennot ja helppo palvelun valinta. Palvelua suunniteltaessa on huomioitava siirrettävän viestin pituuden rajoitukset molemmissa siirto-

suunnissa sekä helppo interaktio. Helppo interaktio tarkoittaa tässä yhteydessä esimerkiksi sitä, että edestakaisin lähetettävien viestien määrä on pieni ja että palvelut, niiden parametrit ja linkit on nimetty intuitiivisesti helposti muistettaviksi. Lisäksi vaaditaan turhien interaktioon liittyvien, uutta oppimista vaativien muutoksien välttämistä palvelun käyttöönoton jälkeen.

Princess-järjestelmä voidaan konfiguroida tunnistamaan tekstiviestikäyttäjä joko lähettäjän numeron tai käyttäjätunnuksen ja salasanan perusteella. Näiden valinnassa ovat vastakkain tietoturva ja sisäänkirjautumisen helppous: lähettäjän numero on helppo tapa, koska GSM-järjestelmä liittää sen viestiin automaattisesti. Toisaalta se ei estä luvatonta käyttöä, jos asiaankuulumaton käyttäjä pääsee vapaasti käsiksi päätelaitteeseen. Lisäksi lähettäjän numeron väärentäminen on joissain tapauksissa mahdollista kytkeytymällä suoraan tekstiviestikeskukseen.

Järjestelmä voi avustaa palvelun valintaa useilla eri tavoilla: Helpointa on, jos käyttäjän ei tarvitse valita palvelua ollenkaan. Tämä voidaan toteuttaa siten, että järjestelmään määritellään oletussivu, joka sisältää oleellisen informaation. Esimerkiksi etusivuksi voidaan määritellä jääkiekko-ottelun tulos, jonka käyttäjä saa lähettämällä tyhjän viestin määrättyyn numeroon. Toisaalta tämä ei enää riitä, jos saman numeron kautta on pystyttävä käyttämään useita palveluja. Tällöin käyttäjä kirjoittaa viestiin palvelun nimen lyhenteen, joka on joko palveluntarjoajan määrittämä tai oletuksena kahdeksan ensimmäistä merkkiä koko nimestä (kuva 19).



Kuva 19. Palvelujen konfigurointi-ikkuna.

Taulukko 6 sisältää esimerkin testipalvelun käytöstä tekstiviesteillä. Testipalvelu on yksinkertaisin mahdollinen, sillä se näyttää ainoastaan saapuneen pyynnön ja yhden linkin, minkä vuoksi sen avulla voi testata eri ominaisuuksia monipuolisesti. Koska

esimerkissä käyttäjän tunnistus tapahtuu puhelinnumeron perusteella, ensimmäisessä viestissä ei tarvitse olla käyttäjätunnusta ja salasanaa.

Taulukko 6. Esimerkki palvelun käytöstä tekstiviesteillä.

Viesti	Vastaus
(tyhjä viesti)	Send name of service: anothert or internet
anothert	Address: princess://AnotherTestservice/. This is a link ($link1)
link1	Address: princess://AnotherTestservice/link_target/
anothert &param arvo	Address: princess://AnotherTestservice/?param=arvo
/ali/doc &param arvo2	Address: princess://AnotherTestservice/ali/doc?param=arvo2

Käyttäjän kannalta tärkeimpiä ja helpoimpia toimintoja ovat palvelun valinta ja linkkien seuraaminen. Linkit voidaan tunnistaa sulkuihin merkityn $-merkin ja linkin nimen perusteella. Jos linkillä ei ole nimeä, järjestelmä luo sen automaattisesti.

Palvelun valinnan ja linkkien seuraamisen lisäksi on mahdollista tehdä myös monimutkaisempia pyyntöjä, joissa viitataan suoraan alipalveluihin tai välitetään parametreja. Parametrien avulla voidaan periaatteessa käyttää XHTML-lomakkeita vaikka niitä ei nykyisellään esitetäkään mitenkään viesteissä. Esittäminen olisi mahdollista samalla tavalla kuin linkkienkin (esimerkiksi suluissa voisi olla &-merkki ja muuttujan nimi), mutta tämä lisäisi käyttäjän muistettavien asioiden määrää ja huonontaisi siten käytettävyyttä.

Käyttäjän interaktiota palveluun voisi laajentaa ja helpottaa vielä ottamalla käyttöön paremmat oikopolut sekä kehittämällä linkkien ja palvelujen nimeämistä älykkäämmäksi. Oikopolkujen avulla voitaisiin viitata helpommin palvelusta useimmin haettavaan sisältöön. Nimeämistä voitaisiin kehittää esimerkiksi siten, että palvelun tai linkin nimeksi valittaisiin aina lyhin mahdollinen yksilöivä merkkijono – tosin mahdollisimman lyhyt ei ole aina helppo muistaa.

Vastaukseen otetaan niin paljon tekstiä kuin yhteen viestiin mahtuu jättämällä palveluntarjoajan merkitsemien prioriteettien avulla vähemmän merkityksellisiä osia pois. Samaan tarkoitukseen voisi käyttää myös HTML-elementtejä esimerkiksi olettamalla, että oleellinen sisältö on ylemmän tason otsikoissa ja kappaleen ensimmäisessä lauseessa. Tämän menetelmän luotettavuus olisi kuitenkin huonompi kuin valitun.

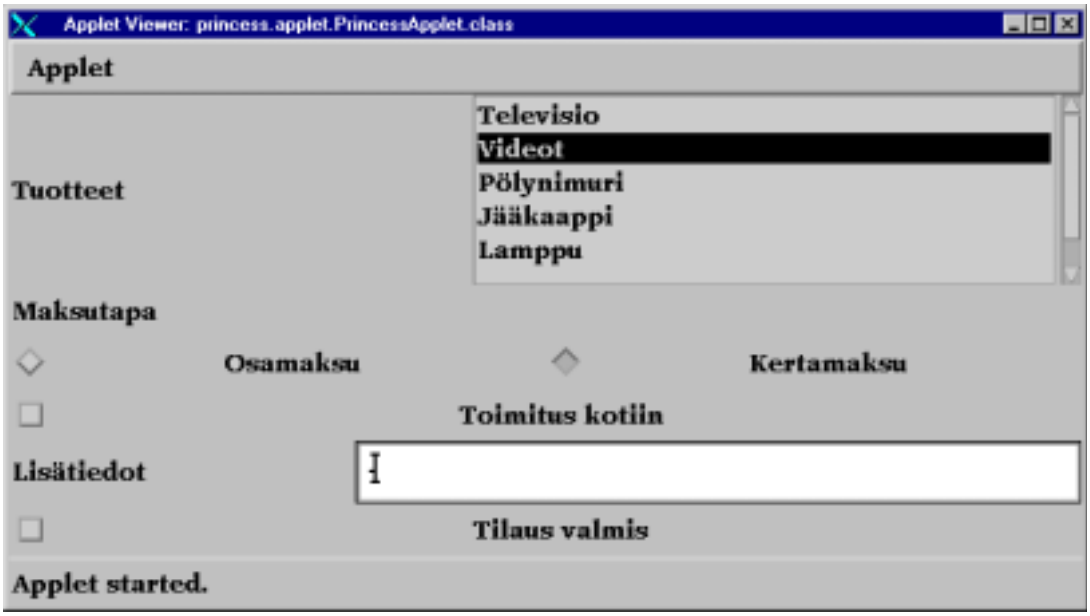
Palvelusta uloskirjautuminen tapahtuu automaattisesti, jos käyttäjä ei ole tiettynä aikana aktiivinen. Ongelmana on vain viipeen pituuden määrittäminen, jotta yhteys ei katkea käyttäjän kannalta ennenaikaisesti.

5.3.3 Sähköposti

Sähköpostiyhdyskäytävä toteutettiin vain yksisuuntaisena, jolloin sitä voidaan käyttää tilattujen ilmoitusten välittämiseen palvelusta käyttäjälle. Myös kaksisuuntainen toteutus olisi mahdollinen komentopohjaisena samaan tapaan kuin tekstiviesteissäkin. Tämän toteuttamista ei kuitenkaan pidetty oleellisena, koska jos käyttäjä voi lukea sähköpostinsa, niin hänellä on yleensä myös mahdollisuus käyttää ainakin tekstipohjaista selainta.

5.3.4 Java-sovellus

Projektin ensimmäisessä vaiheessa toteutettiin Java-appletti (GUIWriter). Tarkoituksena oli hajauttaa laskentaa ja älykkyyttä asiakaspäähän. Toteutuksessa on kuitenkin puutteita: palvelujen ulkoasusta oli vaikea tai mahdotonta saada siisti ja toimiva (kuva 20) aiemmin luvussa 5.2 mainittujen käyttöliittymämallin puutteiden takia.



Kuva 20. Testipalvelu Java-appletin kautta katsottuna.

Toisessa vaiheessa Java-yhdyskäytävää ei enää toteutettu abstraktin käyttöliittymämallin muutoksen jälkeen. Koska tällöin valittiin uudeksi malliksi XHTML, olisi yhdyskäytävä ollut käytännössä WWW-selain, joita on markkinoilla muutenkin.

5.3.5 Puheohjaus

Osittain Princess-projektissa kehitetty suomen kielen puheentunnistin kykenee tunnistamaan yksittäisiä sanoja dynaamisesti konfiguroitavasta, suppeasta sanastosta. Tunnistus tapahtuu mallien perusteella, joita ovat 16 suomen kielen äännettä sekä hiljaisuus, hengitys ja loppukonsonantit. Toteutusta on helpottanut se, että toisin kuin englannin kielessä, suomessa muutamia poikkeuksia lukuun ottamatta äänteet vastaavat kirjoitettuja merkkejä. Puheentunnistinta on opettanut ainoastaan yksi henkilö, joten se ei ole puhujariippumaton ennen kuin puhujien määrää on lisätty. (Peltola et al. 1999.)

Puheentunnistusmoduuli integroitiin ensimmäisessä vaiheessa edellä mainittuun Java-pohjaiseen käyttöliittymään. Liitos on toteutettu dynaamisesti käyttöliittymän komponenteista muodostettavan tilakoneen avulla. Siinä päätilojen eli eri komponenttien välillä navigoidaan yksinkertaisilla "seuraava" ja "edellinen" -komennoilla. Alitilojen eli komponenttien tilojen välillä siirtyminen riippuu komponentista itsestään: lista-komponentissa valinta voi joko tapahtua "ylös", "alas" ja "valitse" -komennoilla tai sanomalla suoraan valittavan alkion nimi. Painikkeiden käyttöön riittää "valitse"-komento. Vapaan tekstin syöttäminen ei onnistu, koska puheentunnistin ei ymmärrä jatkuvaa puhetta ja tukee ainoastaan suppeaa sanavarastoa. Tämän takia puheohjaus rajoittuu niihin operaatioihin, jotka voidaan normaalisti hoitaa hiirellä valitsemalla. (Peltola et al. 1999.)

Toisen vaiheen puheliittymän toteutusta ei olla vielä tätä kirjoitettaessa aloitettu. Vaihtoehtoina on liittää puheentunnistin johonkin olemassa olevaan selaimeen. Tämä voi tapahtua lähdekoodeja muuttamalla, jos ne ovat saatavilla. Toinen vaihtoehto on tehdä liitos vammaisia varten suunniteltuja rajapintoja käyttämällä (esimerkiksi Java Accessibility API). Toisaalta olisi mahdollista kokeilla myös uusia XML-pohjaisia dialoginkuvauskieliä, kuten VoiceXML:ää (VoiceXML-- 1999), jolloin lopputuloksena olisi yksinkertaisimmalla päätelaitteella, puhelimella, käytettävä toteutus.

5.3.6 Yhteenveto ja arviointi

Taulukoon 7 on koottu yhdyskäytävien toteutuksesta saadut kokemukset. Se osoittaa, että käyttöliittymää adaptoiva järjestelmä on toteutettavissa ainakin valitulla päätelaitejoukolla.

Taulukko 7. Käyttöliittymämallin adaptoinnin taso eri muotoihin projektin ensimmäisessä ja toisessa vaiheessa: 0=ei adaptointia, 1=konversio, 2=skaalaus, 3=optimointi (ks. s. 45).

	1. Vaihe	2. Vaihe
WWW-selaimet	1	0
Tekstiviestit	3	3
Sähköposti	– (ei toteutettu)	– (ei interaktiivinen)
WAP-selaimet	– (ei toteutettu)	2? (pääosin toteuttamatta)
Graafinen käyttöliittymä (Java-sovellus)	1	– (ei toteuteta)
Puheohjaus graafiseen käyttöliittymään	2	– (ei toteutettu)
Puheohjaus ilman graafista käyttöliittymää	– (ei toteutettu)	3? (ei toteutettu)

Lopputuloksen toimivuutta todellisessa käyttöympäristössä on vaikea arvioida, sillä suunnitteluun ei ole osallistunut todellisia käyttäjiä eikä käytettävyystestejä olla tehty. Käytettävyysheuristiikkoja on kyllä sovellettu suunnittelussa, mutta niillä ei saada kaikkia ongelmia esille – varsinkaan arviointiin osallistuvien henkilöiden määrän ollessa pieni (Nielsen 1993, 155–156).

Toteutusta voidaan arvioida myös palvelujen näkökulmasta. Projektin kuluessa yliopiston Mediateam kehittää esimerkkipalveluja, joilla voidaan testata ja demonstroida järjestelmää. Tosin rajoituksena on, että nykyisiä palveluja on kehitetty samanaikaisesti muun järjestelmän kanssa, joten niiden suunnittelussa ei ole voitu ottaa alusta asti huomioon muun järjestelmän kehityksessä syntynyttä tietämystä. Tähän mennessä projektissa on kehitetty useita informaatiopalveluita sekä videovalvontapalvelu.

Informaatiopalvelut

Ensimmäisessä vaiheessa toteutettiin informaatiopalveluja, joiden kautta oli mahdollista hakea tietoja pörssikurssi-, henkilö- ja tiesäätietoja. Nämä ovat hyviä demonstraatioita

liikkuvuuden suhteen, sillä niitä on mahdollista ja tarpeellista käyttää yksinkertaisimmilla ja liikkuvimmilla päätelaitteilla. Palvelut toimivat parhaiten WWW-selaimella ja tekstiviesteillä. Sen sijaan Java-käyttöliittymän kanssa oli ongelmia aiemmin mainittujen abstraktin käyttöliittymämallin ongelmien takia. Puheentunnistus oli myös ongelmallinen, sillä palvelut oli toteutettu pääosin englannin kielellä, mikä vaati käyttäjältä erikoista tekstin ääntämistä, koska puheentunnistin on toteutettu suomen kieltä varten.

WWW-palvelut

Järjestelmään on otettu myös mahdollisuus käyttää olemassa olevia WWW-palveluita. Tämä toimii siten, että käyttäjän osoittama sivu haetaan HTTP-protokollalla, muutetaan ulkopuolisella työkalulla XHTML-muotoon ja adaptoidaan tämän jälkeen haluttuun päätelaitteeseen. Riippuu pitkälti WWW-palvelun toteutuksesta, kuinka toimiva tämä menettely on. Alla on listattu joitakin rajoittavia tekijöitä:

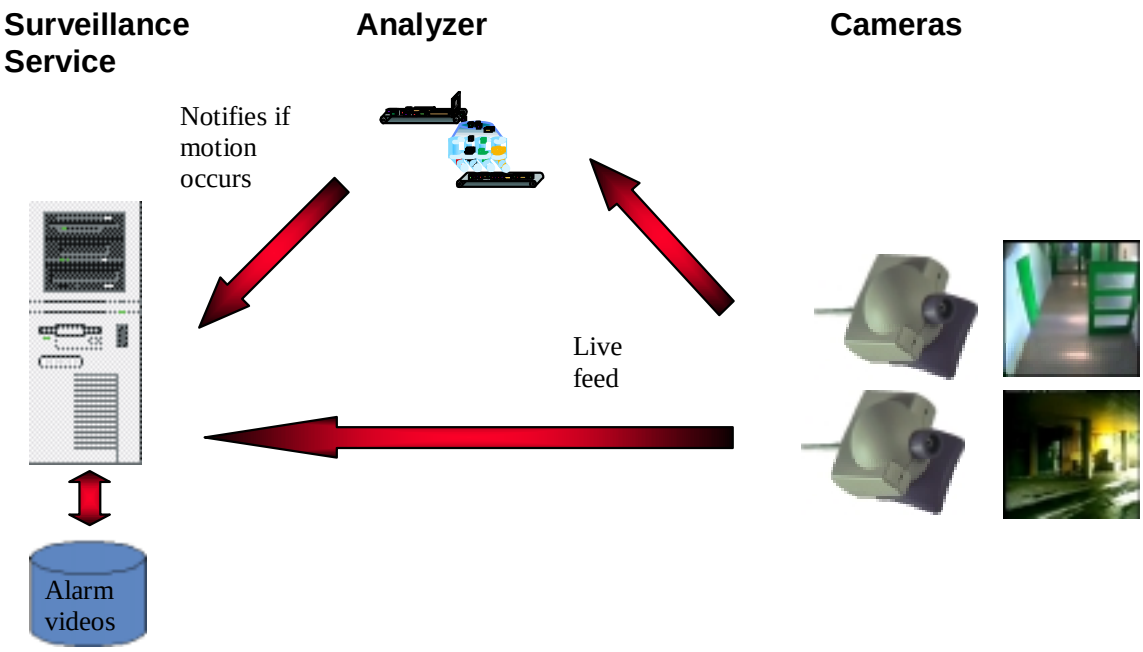
- *Virheelliset tavat käyttää HTML-kieltä.* Esimerkiksi kuvien ja tekstin asemointiin käytetyt taulukot pitäisi voida poistaa, mutta niitä on vaikea erottaa oikeista taulukoista.

- *Semanttisen priorisoinnin puute.* Sivuilta pitäisi yleensä poistaa vähemmän merkityksellisiä osia, jotta lopullinen esitys mahtuisi esimerkiksi tekstiviestiin. Tämä taas ei onnistu nykyisellään automaattisesti.

- *Skriptien, applettien yms. käyttö.* Jos palvelut vaativat toimiakseen selainpäässä toimivia ohjelmia, niitä ei voida adaptoida.

Videovalvontapalvelu

Toisessa vaiheessa toteutettiin videovalvontapalvelu (kuva 17; kuva 21). Palvelulla voi seurata reaaliaikaista videokuvaa useasta eri kamerasta tai tilata hälytykset liikkeentunnistimen havaitsemasta liikkeestä. Tämä palvelu ei demonstroi niinkään adaptiivisuutta vaan pikemminkin eri päätelaitteiden käyttöä eri tarkoituksiin, sillä esimerkiksi videon siirto onnistuu nykyisellään vain työpöytäkoneisiin. Palvelu toimii parhaiten siten, että järjestelmää pyydetään antamaan havaitusta liikkeestä tekstiviesti- tai sähköpostihälytykset, joiden perusteella vartija voi siirtyä katsomaan tehokkaammalla päätelaitteella järjestelmän tallentamia avainkuvia tai videota.



Kuva 21. Videovalvontapalvelu (Forstadius & Löytynoja 1999).

5.4 Toteutuksessa käytetyt työkalut

5.4.1 Java

Järjestelmä on toteutettu kokonaan Java-ohjelmointikielellä. Se on ainakin tässä projektissa saatujen kokemuksien perusteella erittäin käyttökelpoinen valinta sekä ohjelmointikielenä että arkkitehtuurin perustana. Ohjelmointikielenä se on helppo ja mukana tulevat luokkakirjastot ovat käyttökelpoisia ja toteutusta helpottavia – esimerkiksi hajautusratkaisu on helppo tehdä Java RMI:n avulla. Myös alustariippumattomuus tuli testattua, sillä järjestelmää ajettiin ja kehitettiin monessa eri käyttöjärjestelmä- ja laiteympäristössä (Windows 95/NT, Solaris, Linux). Negatiivinen piirre oli Java 2 -työkalujen, muun muassa debuggereiden, kehittymättömyys.

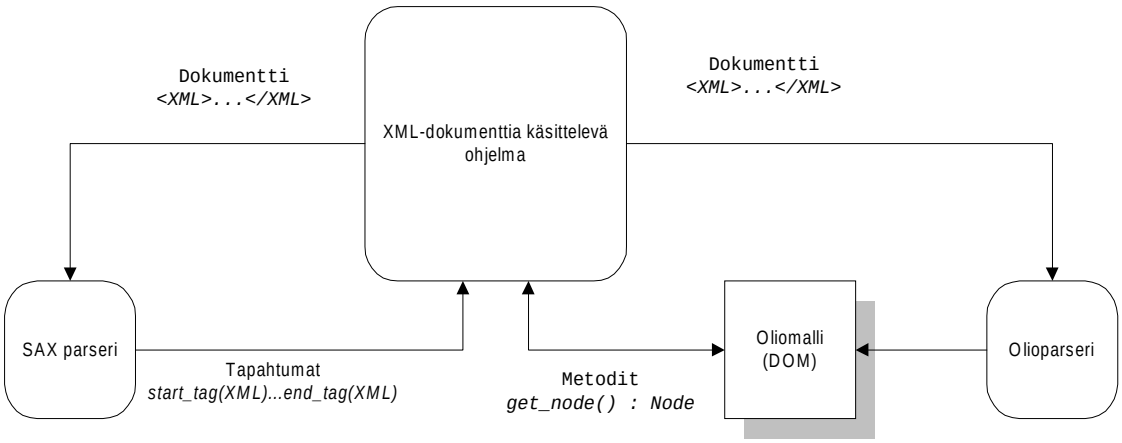
Käytetty Java-versio oli ensimmäisessä vaiheessa Java 1.1 ja toisessa Java 2. Perusominaisuuksien lisäksi käytettiin Servlet API:a HTTP-yhdyskäytävän (WWW/WAP) toteuttamiseen ja Comm API:a sarjaportin kautta tapahtuvaan GSM-korttipuhelimen käsittelyyn SMS-yhdyskäytävässä.

5.4.2 XML

Tässä kappaleessa kuvataan lyhyesti Princessissä käytettyjä XML-työkaluja ja arvioidaan niiden käyttökelpoisuutta.

Parserit

XML-parserit ovat yleisiä työkaluja, jotka lukevat tekstimuotoisia XML-dokumentteja ja generoivat siitä joko oliomallin tai kutsuvat tapahtumankäsittelijän metodeja dokumentin rakenteen mukaisesti (kuva 22). Parseri voi myös tarkistaa sen, että dokumentti on validi eli että se vastaa annettua dokumenttityyppiä.



Kuva 22. Parserityypit ja niiden käyttö sovelluksessa.

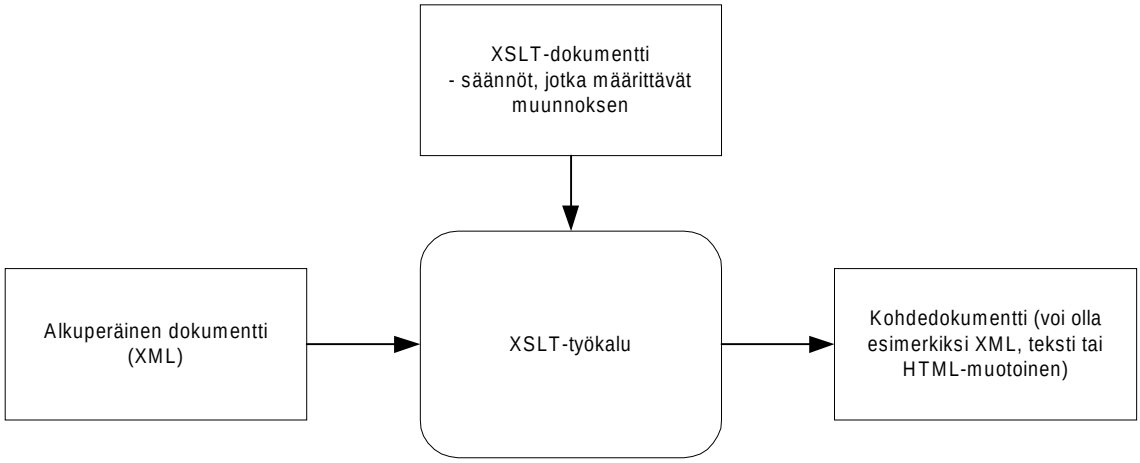
Princessissä käytetään sekä olio- että tapahtumapohjaisia SAX-parsereita. Viimeksi mainittujen käyttöalue on rajoittuneempi, mutta ne soveltuvat erittäin hyvin suoraviivaiseen käsittelyyn – eli tilanteeseen, jossa dokumentti käsitellään alusta loppuun ilman että käsittelykohdasta tarvitsee käsitellä viittauksia eteen- tai taaksepäin. Princessissä niitä käytettiin esimerkiksi dokumentissa olevien linkkien muunnoksiin järjestelmän sisäisen muodon ja Internetissä käytetyn välillä.

DOM on standardoitu ohjelmointikieliriippumaton oliorajapinta, joka sisältää XML- tai HTML-dokumenttien luomiseen, selaamiseen tai muokkaamiseen tarvittavat metodit (W3C 1998b). Alkuperäinen käyttökohde DOM:lle oli dynaamisten WWW-sivujen luominen, mutta nykyään sen käyttöalue on laajentunut. Sen avulla voidaan tehdä kaikki ne operaatiot, jotka voidaan tehdä tapahtumapohjaisesti tai jäljempänä esiteltävillä muunnostyökaluilla. Toisaalta tämä ei ole aina järkevää, sillä rajattuun käyttötarkoitukseen suunnitellut työkalut ovat yleensä tehokkaampia ja helpompia käyttää.

Muunnostyökalut

Muunnoksia XML-dokumenttityypistä toiseen tai muihin kuin rakenteellisiin kieliin voi tehdä kirjoittamalla ohjelma, joka käyttää apuna edellä kuvattuja parsereita. Toisaalta on mahdollista käyttää erityisiä muunnostyökaluja. Näitä työkaluja on olemassa yleisiä ja vain tietylle dokumenttityypille suunniteltuja.

XSLT on geneerinen tapa tehdä muunnoksia XML:n sovelluksesta toiseen (kuva 23). Se on XML:n sovellus, jolla voi kuvata mallipohjaisesti säännöt, joiden perusteella XSLT-työkalu luo lähdedokumentista kohdedokumentin (W3C 1999c). Princessissä käytetään XSLT-työkaluja monimutkaisempien, dokumentin rakennetta koskevien, muunnosten tekemiseen.



Kuva 23. XSLT:n käyttö XML-muunnoksien tekemiseen.

Muita muunnostyökaluja ovat ainakin DSSSL (SGML-muunnostyökalu, joka vastaa perusperiaatteiltaan XSLT:tä) sekä lukuisa joukko vain tietyn dokumenttityypin muunnoksen osaavia työkaluja. Viimeksi mainittu ryhmä osaa käsitellä yleensä jonkin tekstinkäsittelyohjelman tiedostomuotoa.

Esitystyökalut

XML-elementeillä, toisin kuin HTML-elementeillä, ei ole yleisesti määriteltyä semantiikkaa, joten niitä ei voida esittää samalla tavalla. Yleisesti XML-dokumentin esittäminen voi tapahtua kiinnittämällä kullekin elementille visuaalinen tai joku muu esitys. Tästä tavasta on esimerkkinä CSS (kuva 6). Toinen mahdollisuus on konvertoida dokumentti johonkin toiseen muotoon, jonka semantiikan esitykseen käytetty ohjelma ymmärtää. Tällaisia tunnettuja muotoja ovat ainakin HTML, XSL ja PDF.

Princessissä käytetty XML:n sovellus on XHTML. Siksi miltei jokainen HTML-selain osaa näyttää sitä adaptoimatta sellaisenaan.

5.5 Muut adaptoivat järjestelmät

Aloitettaessa Princess-projektia vuonna 1998 ei vielä ollut tiedossa muita vastaavia järjestelmiä. Sen jälkeen on kuitenkin tullut useita julkaisuja mm. WAP:n myötä. Tässä luvussa esitellään kaksi ilmeisesti pisimmällä olevaa adaptoivaa järjestelmää. Pois on jätetty ratkaisut, jotka toimivat vain tietyn laitteen, protokollan tai ohjelmiston kanssa tai joiden sovellusalue on muuten suppea (esimerkiksi WAP-konversiot tai mainosten poistaminen WWW-sivuilta).

Oracle Portal-to-Go (Panama)

Portal-to-Go ei ole vielä valmis tuote, mutta ennakkotietojen mukaan sen tavoitteena on mahdollistaa olemassa olevan Internet-sisällön käyttö langattomista päätelaitteista. Järjestelmä konvertoi automaattisesti verkossa olevaa HTML- ja XML-sisältöä kunkin päätelaitteen tukemalle kuvauskielelle, joista ovat esimerkkeinä HTML, WML, sekä luonnollisen puheen kuvauksessa käytettävä VoxML. Sisältöä voi tuoda Internetin lisäksi myös tietokannoista.

Portal-to-Go sisältää työkalun, jolla analysoidaan palveluksi otettava WWW-sivusto. Jos kyseessä on yksinkertainen informaatiopalvelu, niin analysoinnin voi suorittaa käyttäjä, mutta monimutkaisemmat interaktiiviset järjestelmät jäävät ylläpitäjien tehtäviksi. Lisäksi järjestelmään kuuluu työkalu, jolla käyttäjä voi tavallisen selaimen avulla määritellä itselleen henkilökohtaisen portaalin käyttämistään palveluista. Tämä helpottaa langattomien päätelaitteiden käyttöä rajoittamalla valintojen lukumäärää, jotka näkyvät palveluun kytkeydyttäessä. Palvelut voidaan konfiguroida myös siten, että Portal-to-Go hoitaa automaattisesti käyttäjän puolesta usein tarvittavien tietojen, kuten käyttäjätunnuksien, syöttämisen.

Muita Portal-to-Go:n ominaisuuksia ovat push-tuki, eli mahdollisuus välittää viestejä asynkronisesti palvelusta tilaajalle, sekä rajapinnat laskutuksen ja järjestelmänvalvonnan hoitamiseen (SNMP). Laskutus on mahdollista eriyttää käyttäjäryhmän ja informaation arvon perusteella. (Oracle-- 1999.)

Spyglass Prism

Spyglass Prism on vanhempi järjestelmä kuin Oracle Portal-to-Go. Uudemmissa versioissa on WAP-tuki, mutta alunperin se on suunniteltu nopeuttamaan WWW:n käyttöä PDA-laitteissa. Sen ominaisuuksiin kuuluu kuvien datamäärän adaptointi pienemmäksi sekä käytetyn kuvauskielen konvertointi päätelaitteen ominaisuuksia vastaavaksi. Viimeksi mainittu tarkoittaa esimerkiksi Java-applettien poistoa. Käytettävyyttä pienillä näytöllä varustetuissa laitteissa voi parantaa myös valintatyökalun avulla, jolla merkitään tärkeimmät osat WWW-sivusta näytettäväksi kyseisillä laitteilla.

Muita tuotteen ominaisuuksia ovat tietyn sisällön suodattaminen pois lapsikäyttäjiltä, tietokantarajapinta esimerkiksi laskutuksen toteuttamiseen sekä välimuisti alkuperäisille ja adaptoiduille kuville. Jatkossa Spyglass on julkaissut kehittävänsä yhdessä Lucentin kanssa yksinkertaisella puheohjauksella toimivaa järjestelmää, jota voidaan käyttää tavallisella puhelimella. (Spyglass-- 1999.)

6. Lopuksi

Tässä viimeisessä luvussa esitetään tiivistetysti alatutkimusongelmittain vastaus tutkimusongelmaan sekä esitetään jatkotutkimusideoita yhdessä tuloksen arvioinnin kanssa.

6.1 Tutkimusongelma

Tutkielmassa on pyritty vastaamaan tutkimusongelmaan "Mitä haasteita käyttöliittymän adaptointi eri päätelaitteisiin ja verkkoihin tuo, ja miten niihin voidaan vastata?"

Adaptiivisuudelle asettavat vaatimuksia päätelaitteiden ja niiden ohjelmistojen ominaisuudet. Lisäksi täytyy ottaa huomioon muut ihmisen ja koneen vuorovaikutukseen liittyvät tekijät, kuten käytettävyys eri oloissa sekä hajautetussa ympäristössä myös verkojen ja verkkoprotokollien ominaisuudet, kuten yhteyden taso, tietoturva, liikkuvuuden tuki ja laskutustapa.

On mahdollista rakentaa adaptoiva järjestelmä, joka ottaa automaattisesti huomioon edellä mainittuja tekijöitä. Tämä aiheuttaa myös vaatimuksia palvelujen kehittäjille, koska palvelut täytyy kuvata tavalla, joka on adaptoitavissa eri päätelaitteisiin (abstrakti käyttöliittymämalli). Toisaalta täytyy myös hyväksyä se, että kaikkien päätelaitteiden kaikkia rajoituksia ei voida kiertää ulkopuolisilla järjestelmillä kehittämättä itse laitetta.

Seuraavassa vastataan tutkimusongelmaan tarkemmin käymällä läpi alatutkimusongelmat.

Millaisia käyttöliittymäratkaisuja ja palvelutyyppejä eri päätelaitteet tukevat?

Päätelaitteiden kirjo on suuri: esimerkiksi Princess-projektissa se kattaa laitteita taskukokoisista puhelimista pöydällä pidettäviin työasemiin. Niiden luokittelu on vaikeaa, sillä esimerkiksi puhelinten ja taskutietokoneiden raja on hämärtymässä, ja toisaalta käytetty termistö on kirjavaa. (Luku 3.)

Princess-projektissa tutkitaan pääosin nykyisin yleisesti käytössä olevia päätelaitteita. Niinpä tuettuja sovellustyyppejä tai käyttötapoja ovat puheohjaus, tekstiviestit, sähköposti, WAP- ja WWW-selainsovellukset sekä Java-työpöytäsovellukset. Informaatiopalvelut, kuten aikataulut, sääennusteet jne., ovat hyviä esimerkkejä palvelutyypistä, joka on käytettävissä kaikilla edellä mainituilla tavoilla. Toisaalta joissain tapauksissa tarjottavan sisällön esitystapa, kuten video, voi rajoittaa päätelaitteiden joukkoa. Lisäksi

protokollatason erot aiheuttavat sen, että jotkut palvelut sopivat joillekin päätelaitteille toisia paremmin: hyvä esimerkki on tekstiviestien ja selainsovellusten erot, sillä ensin mainitut sopivat ilmoitusten ja hälytysten välittämiseen ja jälkimmäisen interaktiiviseen käyttöön. (Luvut 5.3.1–5.3.6.)

Mikä on adaptiivinen käyttöliittymä ja miten käyttöliittymää voidaan adaptoida?

Adaptiivisuudella tarkoitetaan kykyä sopeutua johonkin tekijään. Tässä tutkielmassa sillä tarkoitetaan kykyä sopeutua eri päätelaite- ja verkkoympäristöihin.

Adaptointi voi tapahtua niin suunnittelun ja toteutuksen kuin käytönkin aikana. Ensin mainittu tarkoittaa sitä, että järjestelmä suunnitellaan ja toteutetaan niin, että se toimii ennalta määrätyssä joukossa eri mahdollisuuksia – tässä tapauksessa tietyssä päätelaitejoukossa. Käytönaikaisen adaptoinnin lähtökohta on, että suunnitteluvaiheessa ei voida ottaa kaikkea huomioon.

Käytön aikaisen adaptoinnin perusedellytyksiä ovat adaptoiva järjestelmä ja kuvauskieli, jolla voidaan kuvata adaptoitava materiaali. Abstraktilla käyttöliittymämallilla tarkoitetaan käyttöliittymän kuvaustapaa, joka voi olla esitystapariippumaton tai esitystavan suhteen redundanttinen tai molempia. Redundanttisuus merkitsee tässä sitä, että esityksessä on useita vaihtoehtoisia tapoja esittää sama asia, joista valitaan kuhunkin tilanteeseen sopivin. (Luku 4.)

Miten adaptoiva järjestelmä voidaan rakentaa ja mitä työkaluja voidaan käyttää?

Princess on järjestelmä, joka adaptoi verkkopalvelun käyttöliittymän ja sisällön eri päätelaitteisiin. Järjestelmään kuuluu komponentteja, jotka adaptoivat abstraktin käyttöliittymäkuvauksen kutakin päätelaitetta varten sekä vastaavat siitä, että esitys saadaan siirrettyä päätelaitteeseen ja että interaktio käyttäjän ja Princess-järjestelmän välillä toimii mahdollisimman hyvin ja helposti ottaen huomioon päätelaitteen rajoitukset. (Luvut 5.1 ja 5.3.) Ensimmäisessä vaiheessa käytetty abstrakti käyttöliittymämalli oli itse kehitetty oliomalli, josta kuitenkin luovuttiin. Toisessa vaiheessa valittiin tilalle standardiratkaisu, XML:n sovellus XHTML. (Luku 5.2.) Toisaalta pelkän abstraktin käyttöliittymämallin käyttö ei riitä, jos palvelujen kehittäjät eivät ota tiettyjä rajoituksia huomioon: esimerkiksi resurssit täytyy nimetä niin, että ne ovat helposti muistettavissa tekstiviestejä käytettäessä. (Luku 5.3.)

Tärkeimpiä työkaluja Princess-järjestelmän toteuttamisessa olivat Java- ja XML-teknologiat ja -työkalut. Valintaa voi pitää onnistuneena: XML on hyvä valinta laajennettavuutensa ja laajan työkalutukensa ansiosta. Työkaluista kannattaa mainita erityisesti parserit ja muunnostyökalut. Lisäksi Java ja XML sopivat hyvin yhteen, sillä XML-työkalut on toteutettu usein Javalla. (Luku 5.4.) Myös muut tiedossa olevat adaptoivat järjestelmät perustuvat ainakin osittain XML-teknologioihin – tosin kovin tarkkoja teknisiä yksityiskohtia niistä ei ole saatavilla (Luku 5.5).

6.2 Jatkotutkimus ja tuloksen arviointi

Tämä tutkimus jättää auki joitain kysymyksiä, joihin ei voida vastata näissä puitteissa. Alla on lueteltu joitakin näistä kysymyksistä. Osaan niistä voitaneen vastata jatkossa Princess-projektissa ja osaan muissa tutkimusprojekteissa ja käytännön elämässä toteutettaessa uusia verkkopalveluja.

Tarvitaanko adaptiivisuutta?

Adaptiivisuus poistaa tarpeen ylläpitää useita versioita samasta palvelusta eri päätelaitteita varten, joten sitä voi pitää palvelujen kehittämistä helpottavana ominaisuutena. Toisaalta kiinnostavatko eri päätelaitteiden käyttäjiä sama sisältö ja palvelut? Vaihtoehto voisi olla toteuttaa päätelaitekohtaisia palveluja, jotka perustuisivat päätelaitteen erityisominaisuuksiin, kuten langattomuuteen ja liikkuvuuteen.

Toisaalta tarvitaanko adaptiivisuutta enää tulevaisuudessa vai tekevätkö uudet päätelaite-, käyttöliittymä- ja verkkoteknologiat sen tarpeettomaksi? Esimerkiksi virtuaalitodellisuus on hyvä esimerkki siitä, miten fyysisiä rajoituksia saadaan kierrettyä.

Onko tässä tutkielmassa esitetty tapa toteuttaa adaptiivinen järjestelmä oikea?

Princess-järjestelmässä adaptointi perustuu pitkälti yksittäisten esitysten käsittelyyn. Tämä johtuu käytetystä abstraktista käyttöliittymämallista, jolla palvelu kuvataan esitys (eli sivu tai näyttö) kerrallaan. Toinen mahdollisuus voisi olla abstraktiotason nostaminen korvaamalla käyttöliittymämalli esimerkiksi jonkinlaisella vuorovaikutusmallilla, jolloin voitaisiin adaptoida entistä enemmän vuorovaikutusta kuin yksittäisiä esityksiä. Tätä voitaisiin kutsua termillä IPR – interaction process reengineering (vrt. BPR – business process reengineering). Toisaalta adaptoinnin toteuttaminen tällä tasolla automaattisesti voi olla hankalaa tai jopa mahdotonta.

Toinen kysymys on se, tarvitaanko Princessin kaltaista jonkin verran raskasta ja monoliittistä järjestelmää. Ehkä käyttäjän kannalta parempaan lopputulokseen päästäisiin toteuttamalla adaptiivisuus palvelukohtaisesti valmiiden adaptointimodulien avulla. Tässä suhteessa Princessin kilpailijoita ovat XML ja siihen liittyvät työkalut, joiden avulla on mahdollista toteuttaa erilaisia näkymiä samaan tietoon. Toisaalta tämänkin vaihtoehdon huono puoli on palvelun suunnittelun ja toteutuksen lisääntynyt työmäärä.

Onko Princess-järjestelmän toteutus hyvä?

Princess-järjestelmää ei ole testattu käyttäjän näkökulmasta muuten kuin käytettävyysheuristiikkoja soveltamalla. Tilannetta voisi parantaa esimerkiksi käytettävyystutkimuksella. Tällöin voitaisiin saada vastauksia myös muihin edellä mainittuihin kysymyksiin.

Alla on vielä listattu joitakin Princess-järjestelmään liittyviä seikkoja, joita ei ole juurikaan pystytty huomioimaan tässä tutkielmassa:

- Miten käyttäjien erilaiset kyvyt, tarpeet ja fyysiset ominaisuudet pitäisi ottaa huomioon (adaptoituminen käyttäjän suhteen)?

- Miten laskutuksen pitäisi näkyä käyttäjälle eri päätelaitteilla? Toisaalta laskutuksen pitäisi olla näkymätön, jotta se ei vaikeuttaisi käyttöä, mutta toisaalta taas näkyvä, jotta se ei aiheuttaisi käyttäjälle odottamattomia yllätyksiä.

- Mitä WAP:lla voi tehdä ja mikä on se suhde WWW-tekniikoihin? Esimerkiksi miten WAP-palvelut pitäisi toteuttaa verrattuna nykyisiin verkkopalveluihin?

Teknistä toteutusta ei ole pyritty optimoimaan. Suunnittelupäätöksiä tehtäessä on kyllä otettu huomioon skaalautuvuus, mutta todellista toimivuutta raskaassa käytössä ei ole vielä testattu.

Lähdeluettelo

Benyon, D. 1998. Intelligent Interface Technology: shifting the HCI paradigm in Tutorials at HCI'98 – Intelligent Interface Technology.

Berners-Lee, T. 1998. Web Architecture from 50,000 feet [verkkodokumentti]. [viitattu 30.4.1999]. Saatavissa: http://www.w3.org/DesignIssues/Architecture.html.

Brown, P., Dovey, J. & Chen, X. 1997. Context-Aware Applications: From the Laboratory to the Marketplace. IEEE Personal Communications.Vol 4, nro. 5, s. 58–64.

Burdea, G. & Coiffet, P. 1994. Virtual Reality Technology. New York: John Wiley.

Edwards, W. K. & Mynatt, E. 1994. An Architecture for Transforming Graphical User Interfaces. Proceedings of the ACM symposium on User interface software and technology.

CTI-projektiryhmä. 1999. State-of-the-art in CTI. Oulun yliopisto ja VTT Elektroniikka. Toistaiseksi julkaisematon Tekes-raportti.

Forman, G. & Zahojan, J. 1994. The Challenges of Mobile Computing. IEEE Computer. April 1994, s. 38–47.

Forstadius, J. & Löytynoja, M. Princess Surveillance Service. Presentation slides 22.10.1999.

Grasso, M. A., Ebert, D. S. & Finin, T. W. 1998. The Integrality of Speech in Multimodal Interfaces. ACM Transactions on Computer-Human Interaction. Vol. 5, nro. 4, s. 303–325.

Järvinen, P. & Järvinen, A. 1996. Tutkimustyön metodeista. Tampereen yliopisto.

Kamm, C., Walker, M. & Rabiner, L. 1997. The Role of Speech Processing in Human-computer Intelligent Communication. Appendix of the final report of the NSF workshop on Human-centred systems, Information, Interactivity and Intelligence (HCS). Saatavissa myös: http://www.ifp.uiuc.edu/nsfhcs/talks/rabiner.html.

Korpela, J. 1998. Lurching Toward Babel: HTML, CSS, and XML. Computer. Vol. 31, nro. 7, s. 103–106.

Korva, J. & Metso, M. 1999. Applicability of XML in Princess. VTT Elektroniikka & Oulun yliopisto. Julkaisematon raportti.

Lie, H. W. & Saarela, J. 1999. Multipurpose Web Publishing. Communications of The ACM. Vol. 42, nro. 10, s. 95–101.

Lynch, P. & Horton, S. 1999. Web Style Guide : Basic Design Principles for Creating Web Sites. Yale University Press. Saatavissa myös: http://info.med.yale.edu/caim/manual/.

Marden, P. M. & Munson, E. V. 1999. Today's Style Sheet Standards: The Great Vision Blinded. Computer. Vol 32, nro. 11, s. 123–125.

McKay, S. 1991. CLIM: The Common Lisp Interface Manager. Communications of the ACM. Vol. 34, nro. 9, s. 58–59.

Metso, M., Koivisto, A. & Sauvola, J. 1998. Multimedia Adaptation for Dynamic Environments. Proceedings of the IEEE 1998 Workshop on Multimedia Signal Processing.

Microsoft Accessibility Products. 1999. [viitattu 26.4.1999]. Saatavissa: http://www.microsoft.com/enable/products/default.htm.

Nielsen, J. 1993. Usability Engineering. Lontoo: Academic Press.

Nykysuomen sivistyssanakirja. 1982. 8. painos. Porvoo: WSOY.

Oracle Portal-to-Go, 1999. An Oracle Business White Paper [verkkodokumentti]. [viitattu 28.10.1999]. http://www.oracle.com/mobile/panama/pan_bwp.pdf.

Peltola, J. 1998. Kätkettyihin Markov-malleihin perustuva puheentunnistus. Oulun yliopisto, sähkötekniikan osasto. Diplomityö.

Peltola, J., Plomp, J. & Seppänen T. 1999. A dictionary-adaptive speech driven user interface for a distributed multimedia platform. Euromicro workshop on multimedia and telecommunications conference paper.

Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S. & Carey, T. 1994. Human-computer Interaction. Harlow: Addison-Wesley.

Price, R. 1993. MHEG: An introduction to the future international standard for hypermedia object interchange. ACM Multimedia 93. Julkaisuun viitannut Lennon, J. 1997. Hypermedia Systems and Applications. Berlin: Springer-Verlag.

Rabiner, L, & Juang, B.H. 1993. Fundamentals of Speech Recognition. London: Prentice-Hall.

RFC 2396. 1998. Uniform Resource Identifiers (URI): Generic Syntax. Internet Engineering Task Force (IETF).

Schmidt, A., Beigl, M. & Gellersen, H-W. 1998. There is more to context than location. In Proceedings of the International Workshop on Interactive Applications of Mobile Computing (IMC98), Rostock.

Schwerdtfeger, R. 1998. IBM guidelines for writing accessible applications using 100% pure Java [verkkodokumentti]. [viitattu 22.4.1999]. Saatavissa: http://www.austin.ibm.com/sns/snsjavag.htm.

Spyglass Prism 2.2: Internet Content Delivery Platform. [viitattu 28.10.1999]. Saatavissa: http://www.spyglass.com/solutions/technologies/prism/.

Vanderheiden, G. & Vanderheiden, K. 1992. Accessible Design of Consumer Products [verkkodokumentti]. University of Wisconsin. [viitattu 19.10.1999]. Saatavissa: http://trace.wisc.edu/docs/consumer_product_guidelines/consumer.htm.

Vanderheiden, G. 1997. Anywhere, Anytime (+Anyone) Access to the Next-generation WWW. Computer Networks and ISDN Systems.

WAP Forum 1999. Wireless Markup Language Specification, version 1.1. Saatavissa: http://www.wapforum.org/.

VoiceXML Forum. 1999. [viitattu 19.10.1999]. Saatavissa: http://www.voicexml.org/.

W3C (World Wide Web Consortium) 1998a. Cascading Style Sheets, level 2. W3C Recommendation. Saatavissa: http://www.w3.org/TR/REC-CSS2.

W3C (World Wide Web Consortium) 1998b. Document Object Model (DOM) Level 1 Specification. W3C Recommendation. Saatavissa: http://www.w3.org/TR/REC-DOM-Level-1.

W3C (World Wide Web Consortium) 1998c. HTML 4.0 Specification. W3C Recommendation. Saatavissa: http://www.w3.org/TR/REC-html40.

W3C (World Wide Web Consortium) 1999a. Modularization of XHTML. W3C Working Draft. Saatavissa: http://www.w3.org/TR/xhtml-modularization/.

W3C (World Wide Web Consortium) 1999b. XHTML™ 1.0: The Extensible HyperText Markup Language. W3C Working Draft. Saatavissa: http://www.w3.org/TR/xhtml1.

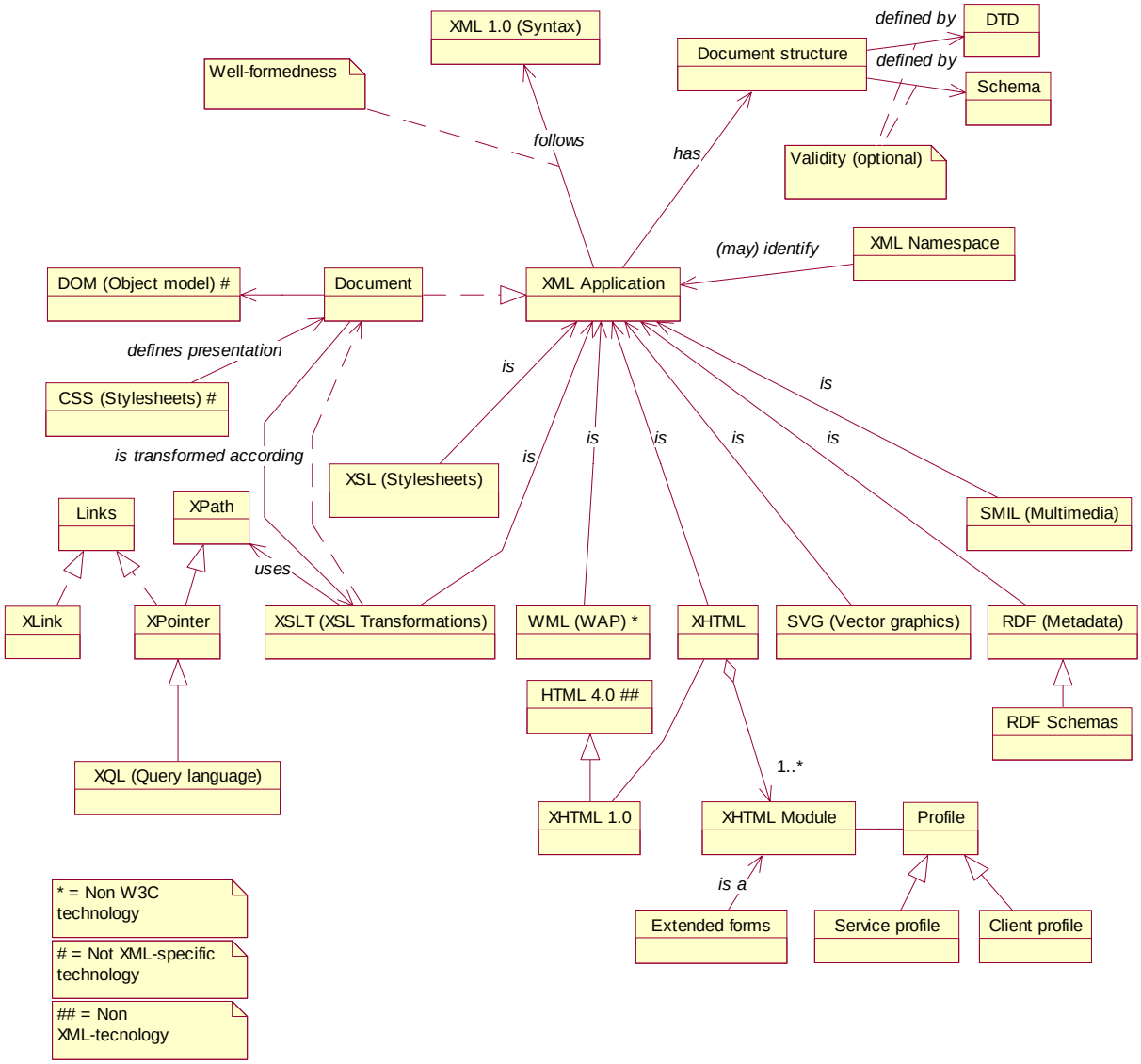
W3C (World Wide Web Consortium) 1999c. XSL Transformations (XSLT). W3C Recommendation. Saatavissa: http://www.w3.org/TR/xslt.

W3C (World Wide Web Consortium) 1999d. HTML Working Group Roadmap. W3C Note. Saatavissa: http://www.w3.org/TR/xhtml-roadmap/.

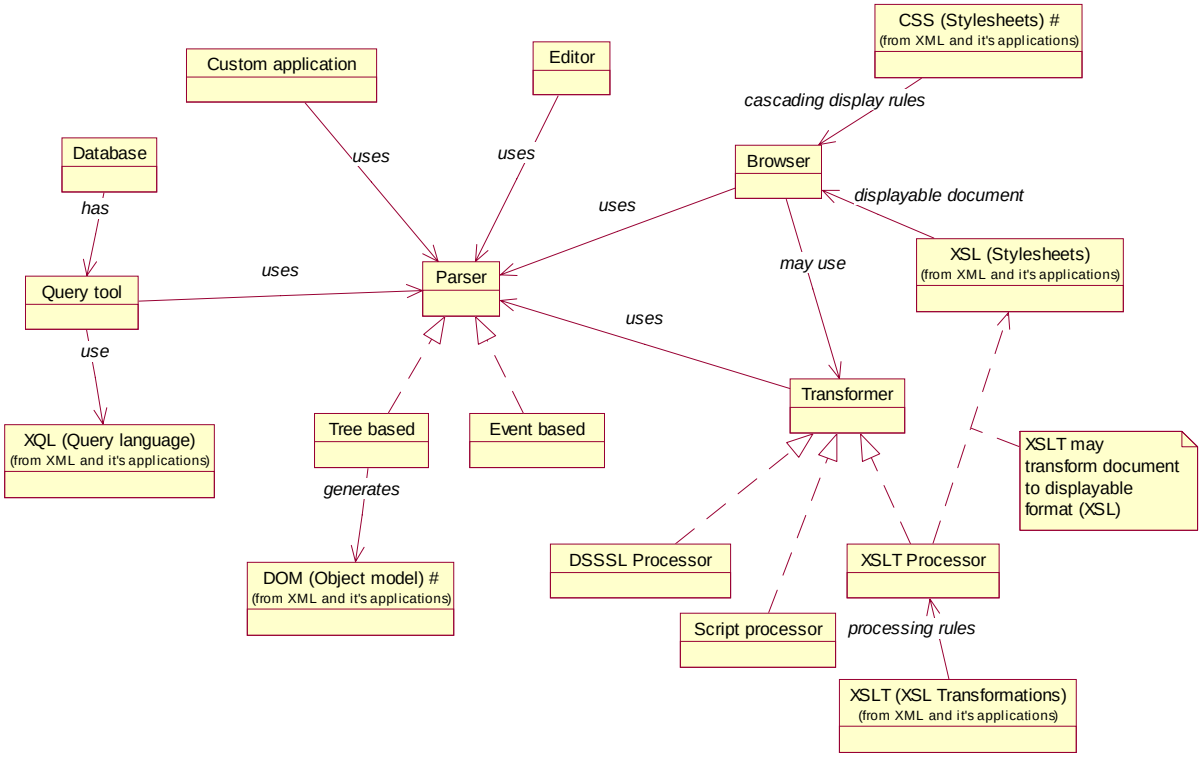
Yao, P. 1999. Internet Programming with Windows CE. Microsoft Internet Developer. No. 5. Saatavana myös: http://www.microsoft.com/mind/0599/webce/webce.htm.

Liite A: XML-käsitekartta

Tässä liitteessä on kuvattu XML-käsitteitä ja niiden riippuvuuksia UML-notaatiota soveltaen. Käsitekartat ovat tulosta projektissa tehdystä XML-selvityksestä. Ensimmäinen diagrammi sisältää XML-peruskäsitteitä ja joitakin sovelluksia sekä jälkimmäinen XML-työkaluja.



Kuva 24. XML-peruskäsitteitä ja -sovelluksia.



Kuva 25. XML-työkaluja.

Liite B: Princess DTD

Princess DTD määrittelee lisäykset, joita on tehty XHTML-dokumenttityyppiin (ks. luku 5.2). Vaikka DTD ei olekaan pakollinen XML-dokumenteissa, se on käyttökelpoinen kehitysaikana toteutusvirheiden estämiseksi.

```
<!--
DTD for Princess Presentation Model.
This DTD module is identified by the PUBLIC and SYSTEM identifiers:

  - PUBLIC "-//Princess//Princess Presentation 1.0//EN"
  - SYSTEM "http://www.ee.oulu.fi/~jpkorva/princess/dtd/princess.dtd">

Note: You must apply a patch to XHTML strict.dtd before using it!

Author: Jari Korva
Version history:
0.1   4.11.1999    First version  (JKO)
-->

<!--================ XML-Namespaces     ===========================-->

<!--
  Note: let's use 'http://www.w3.org/TR/xhtml1' instead of
  'http://www.w3.org/TR/xhtml1/strict' as XHTML namespace
  because this will propably be changed before XHTML becomes a
  Recommendation.
-->
<!ENTITY % XHTML.ns "http://www.w3.org/TR/xhtml1">

<!ATTLIST html
    xmlns:princess      CDATA           #FIXED 'urn:tekes.fi:princess'
>

<!--================ princess:priority  ===========================-->

<!ENTITY % StyleSheet "CDATA">
<!ENTITY % Text "CDATA">

<!ENTITY % coreattrs
 "id                    ID              #IMPLIED
  class                 CDATA           #IMPLIED
  style                 %StyleSheet;    #IMPLIED
  title                 %Text;          #IMPLIED
  princess:priority     CDATA           #IMPLIED"
>
```

```
<!--================ princess:property  ============================-->

<!ELEMENT princess:property EMPTY>
<!ATTLIST princess:property
    name                NMTOKEN         #REQUIRED
    value               CDATA           #REQUIRED
>

<!ENTITY % head.misc
 "(script|style|meta|link|object|princess:property)*"
>

<!--================ princess:price     ============================-->
<!--================ princess:subscribe ============================-->

<!ATTLIST a
    princess:price      CDATA          #IMPLIED
    princess:subscribe  CDATA          #IMPLIED
>

<!ATTLIST form
    princess:price      CDATA          #IMPLIED
>

<!ATTLIST object
    princess:price      CDATA          #IMPLIED
>

<!ATTLIST img
    princess:price      CDATA          #IMPLIED
>

<!--================ XHTML DTD          ============================-->

<!ENTITY % XHTML1-t.dtd PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "strict.dtd">
%XHTML1-t.dtd;
```

Julkaisija

Vuorimiehentie 5, PL 2000, 02044 VTT
Puh. (09) 4561
Faksi (09) 456 4374

Julkaisun sarja, numero ja raporttikoodi

VTT Tiedotteita 2088
VTT-TIED-2088

Tekijä(t)
Korva, Jari

Nimeke

Adaptiivisten verkkopalvelujen käyttöliittymät

Tiivistelmä

Tekesin rahoittamassa Princess-projektissa kehitetään palvelualustaa, jolle voidaan rakentaa missä ja milloin tahansa käytettävissä olevia verkkopalveluja – toisin sanottuna järjestelmää, joka mukauttaa palvelun kulloinkin käytettävissä olevaan päätelaitteeseen. Adaptoinnin kohteina ovat sekä media (esimerkiksi kuvien käsittely tiedonsiirron nopeuttamiseksi) että palvelun käyttöliittymä, jota tarkastellaan tässä pro gradu -tutkielmassa.

Työn lähtökohtana on olemassa oleva tieto käyttöliittymistä: niiden käytettävyys, ihmisen ja koneen vuorovaikutustavat sekä syöttö- ja näyttölaitteet. Näihin liittyvät erilaiset päätelaite- ja sovellustyypit, kuten työasemissa ja mukana kuljetettavissa tietokoneissa toimivat työpöytä- ja selainsovellukset sekä matkapuhelimissa toimivat ääniohjatut sovellukset, tekstiviestit ja WAP-selainsovellukset. Lisäksi tutkielmassa kiinnitetään huomiota tietoliikenneverkkojen ja -protokollien vaikutukseen käyttöliittymäratkaisuihin.

Adaptiivisuuden lisäksi merkittävä käsite on esitystapariippumaton käyttöliittymän kuvaustapa eli abstrakti käyttöliittymämalli, toisin sanottuna kuvaus, josta voidaan adaptoida kussakin päätelaitteessa toimiva käyttöliittymä. Abstraktiksi käyttöliittymämalliksi valittiin projektin ensimmäisessä vaiheessa itse kehitetty oliomalli, joka sisälsi yleisimmät käyttöliittymäkomponentit (esimerkiksi valintalista, linkki, nappi). Toteutuksessa esiintyneiden puutteiden takia alettiin myöhemmin etsiä standardinmukaisempaa ratkaisua, joista lupaavimpana valittiin rakenteellisen kuvauskielen, XML:n, sovellus XHTML.

Tutkielman kohteena oleva Princess-järjestelmän osa on toteutettu Java-kielen ja XML-työkalujen avulla. Tuettuja päätelaite- ja käyttöliittymätyyppejä ovat Java-sovellukset (myös puheohjauksella käytettynä), WWW-selaimet, tekstiviestit ja sähköposti sekä jatkossa myös WAP.

Avainsanat
adaptive user interfaces, universal access, WWW, abstract user interface model

Toimintayksikkö
VTT Elektroniikka, Tietoliikennejärjestelmät, Kaitoväylä 1, PL 1100, 90571 OULU

ISBN			Projektinumero
951–38–5810–3 (URL: http://www.inf.vtt.fi/pdf/)			E0SU00324
Julkaisuaika	Kieli	Sivuja	Hinta
Huhtikuu 2001	suomi, engl. tiiv.	71 s. + liitt. 4 s.	B

Projektin nimi	Toimeksiantaja(t)
Princess	Tekes, Nokia Mobile Phones, Nokia Networks, Sonera, CCC, Kesko

Avainnimeke ja ISSN
VTT Tiedotteita – Meddelanden – Research Notes
1455–0865 (URL: http://www.inf.vtt.fi/pdf/)

Myynti:
VTT Tietopalvelu
PL 2000, 02044 VTT
Puh. (09) 456 4404
Faksi (09) 456 4374

Published by

Vuorimiehentie 5, P.O.Box 2000, FIN-02044 VTT, Finland
Phone internat. +358 9 4561
Fax +358 9 456 4374

Series title, number and report code of publication

VTT Research Notes 2088
VTT–TIED–2088

Author(s)
Korva, Jari

Title

User interfaces for adaptive multimedia services

Abstract

The goal of the Princess project is to research and develop a platform for multimedia services that allows access to the contents regardless of the user's location and the terminal device – i.e. the purpose of the system is to adapt services to match various network and terminal capabilities. This thesis focuses on adapting user interfaces, which constitutes one part of the complete adaptation process.

The starting point of this thesis is the existing knowledge of user interfaces: usability, human-computer interaction types, input/output devices and implementation level technologies for developing user interfaces. Also characteristics of different application types are described, including desktop and browser applications for fixed and mobile terminals as well as short messages and voice-based systems.

In order to achieve adaptivity, a model (i.e. format) that describes the user interface independent of the final presentation modality and platform, is required. This format is called the abstract user interface model or the presentation model – the latter concept is wider because in addition to the user interface, also content needs to be modelled.

An object-based presentation model capable of describing basic UI concepts was developed in the first phase of the Princess-project. Later this model was discarded in favour of a standard-based solution, XHTML, which is a reformulation of HTML in XML.

The Princess system is implemented using Java and XML technologies. Supported terminal/user interface types include WWW and WAP browsers, short messages (SMS) and email. Experiments have been made with graphical Java applications with speech driven input.

Keywords
adaptive user interfaces, universal access, WWW, abstract user interface model

Activity unit
VTT Biotechnology and Food Research, Microbiology and Safety, Biologinkuja 1, P.O.Box 1500, FIN–02044 VTT, Finland

ISBN			Project number
951–38–5810–3 (URL: http://www.inf.vtt.fi/pdf/)			E0SU00324
Date	Language	Pages	Price
April 2001	Finnish, Engl. abstr.	71 p. + app. 4 p.	B
Name of project		Commissioned by	
Princess		Tekes, Nokia Mobile Phones, Nokia Networks, Sonera, CCC, Kesko	
Series title and ISSN		Sold by	
VTT Tiedotteita – Meddelanden – Research Notes 1455–0865 (URL: http://www.inf.vtt.fi/pdf/)		VTT Information Service P.O.Box 2000, FIN–02044 VTT, Finland Phone internat. +358 9 456 4404 Fax +358 9 456 4374	

Markkinoilla olevien päätelaitteiden ja verkkojen kirjo on laajentunut langattomien ja mukana kuljetettavien teknologioiden myötä. Tämä mutkistaa verkkopalvelujen tuottamista: saman sisällön esittäminen useissa eri päätelaitteissa vaatii useiden eri versioiden tekemistä samasta palvelusta.

Princess-projektissa on tutkittu ja kehitetty palvelualustaa, joka huolehtii automaattisesti sisällön esittämisestä eri päätelaitteissa. Projektin osana on syntynyt myös tämä työ, jossa tutkitaan adaptiivisuutta käyttöliittymien näkökulmasta – eli sitä, minkälaisia käyttöliittymäratkaisuja eri päätelaitteissa on ja miten palvelun käyttöliittymä voidaan niissä esittää.

Tätä julkaisua myy
VTT TIETOPALVELU
PL 2000
02044 VTT
Puh. (09) 456 4404
Faksi (09) 456 4374

Denna publikation säljs av
VTT INFORMATIONSTJÄNST
PB 2000
02044 VTT
Tel. (09) 456 4404
Fax (09) 456 4374

This publication is available from
VTT INFORMATION SERVICE
P.O.Box 2000
FIN–02044 VTT, Finland
Phone internat. + 358 9 456 4404
Fax + 358 9 456 4374

ISBN 951–38–5810–3 (URL: http://www.inf.vtt.fi/pdf/)
ISSN 1455–0865 (URL: http://www.inf.vtt.fi/pdf/)