TAMPEREEN AMMATTIKORKEAKOULU OPINNÄYTETYÖ
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

Opinnäytetyö

Tuomas Liimatainen

KOSKETUSNÄYTTÖÄ HYÖDYNTÄVÄ UIQ-SOVELLUS

Työn ohjaaja Lehtori Jari Mikkolainen
Työn teettäjä Sasken Finland Oy, valvojana diplomi-insinööri Jani Turpeinen
Tampere 2009

TAMPEREEN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka
Liimatainen, Tuomas
Kosketusnäyttöä hyödyntävä UIQ-sovellus
Tutkintotyö 44 sivua
Työn ohjaaja Lehtori Jari Mikkolainen
Työn teettäjä Sasken Finland Oy, valvojana diplomi-insinööri Jani Turpeinen
Toukokuu 2009
Hakusanat UIQ, Symbian, käyttöliittymä, kosketusnäyttö

TIIVISTELMÄ

Symbian on älypuhelimissa yleisimmin käytetty käyttöjärjestelmä, johon perustuvista matkapuhelinalustoista yleisimpiä ovat S60- ja UIQ-alustat. S60-alusta on selvästi yleisin Symbianiin perustuva alusta, jota käytetään Nokian älypuhelimissa. Koska kosketusnäytön hyödyntäminen on viime aikoina noussut tärkeäksi kilpailutekijäksi eri laitevalmistajien ja matkapuhelinalustojen välillä, julkaisi Nokia ensimmäisen S60-alustalla toimivan ja kosketusnäyttöä hyödyntävän matkapuhelimen vuonna 2008. UIQ-alusta on alusta alkaen suunniteltu kosketusnäyttöä ajatellen ja esimerkiksi Sony Ericsson on tuonut markkinoille kosketusnäyttöä hyödyntäviä UIQ-puhelimia jo usean vuoden ajan.

Tämän työn tarkoituksena oli perehtyä S60- ja UIQ-alustoihin ja erityisesti kosketusnäytön hyödyntämiseen näillä alustoilla. Alustoihin perehtymisen lisäksi, tehtävänä oli siirtää Sasken Finland Oy:ssä S60-alustalle kehitetty testausympäristö UIQ-alustalle siten, että kosketusnäytön ominaisuuksia hyödynnetään ja että käyttöliittymästä tulee entistäkin kätevämpi. Sovelluksen käyttöliittymä piti toteuttaa siten, että toimintojen suorittaminen on mahdollisimman helppoa ja selkeää kosketusnäytön avulla.

Testausympäristön siirtäminen S60-alustalta UIQ-alustalle saatiin toteutettua suunnitellusti. Käyttöliittymää on paranneltu ja käyttö kosketusnäytön avulla on helppoa ja selkeää.

TAMPERE UNIVERSITY OF APPLIED SCIENCES

Computer Systems Engineering
Software Engineering
Liimatainen, Tuomas
UIQ-application utilizing touch screen
Engineering Thesis 44 pages
Thesis Supervisor Senior Lecturer Jari Mikkolainen
Commissioning Company Sasken Finland Ltd. Supervisor M.Sc. Jani Turpeinen
May 2009
Keywords UIQ, Symbian, user interface, touch screen

ABSTRACT

Symbian is the most wide spread smart phone operating system and S60- and UIQ-platforms are the most common platforms that are based on it. Because touch screen support has recently become very important factor in competition between different mobile phone manufacturers and –platforms, Nokia released it's first S60-platform based phone with touch screen support in 2008. UIQ-platform has been designed to support touch screens from the beginning and Sony Ericsson for example has been selling UIQ-phones with touch screen support for many years.

The purpose of this work was to familiarize oneself with S60- and UIQ-platforms and especially with touch screen usage on them. In addition to getting to know the platforms, a test environment, developed in Sasken Finland Ltd., had to be ported from S60- to UIQ-platform. Touch screen utilization had to be taken into account when porting the application. The application had to have a user interface that is easy to use and takes advantage of the possibilities provided by touch screen.

Porting of the test environment was done as planned. The user interface has been improved and usage of the application via touch screen is easy and clear.

SISÄLLYSLUETTELO

LYHENTEIDEN SELITYKSET

API	Application Programming Interface on ohjelmointirajapinta, jonka kautta eri ohjelmat tai muut yksiköt voivat keskustella keskenään.
Avkon	S60-alustan käyttöliittymäkirjastot kuuluvat Avkon-kehykseen, joka laajentaa UIKON-kirjastoa. Avkon sisältää käyttöliittymään liittyviä komponentteja ja kontrolleja, joista S60-alustan käyttöliittymä muodostuu.
DLL	Dynamically Linked Library on jaettu kirjasto, josta voidaan ajon aikana ladata funktioita suoritettavaksi.
Eikon	UIKON-kehys oli ennen nimeltään Eikon.
EPOC	Symbian-käyttöjärjestelmä oli ennen nimeltään EPOC.
FEP	Front End Processor on yksikkö, joka esikäsittelee dataa ennen sen siirtämistä eteenpäin.
MVC	Model View Controller -arkkitehtuuri on graafisissa sovelluksissa yleisesti käytetty arkkitehtuurityyli, jossa sovellus jaetaan kolmeen osaan: malliin, näkymään ja kontrolleriin.
Qikon	UIQ-alustan käyttöliittymäkirjastot kuuluvat Qikon kehykseen, joka laajentaa UIKON-kirjastoa. Qikon sisältää käyttöliittymään liittyviä komponentteja ja kontrolleja, joista UIQ-alustan käyttöliittymä muodostuu.
SDK	Software Development Kit on kokoelma kehitystyökaluja ja kirjastoja, joiden avulla voidaan kehittää sovelluksia jollekin tietylle alustalle.
Tactile Feedback	Tactile Feedback on tekniikka, jolla käyttäjälle annetaan palautetta kosketusnäytön painalluksista S60-alustalla.
UI	User Interface, käyttöliittymä.
UIKON	Symbian-käyttöjärjestelmän käyttöliittymän kirjastot kuuluvat UIKON-kehykseen, joka on perusta graafisille Symbian-sovelluksille.
WTF	Wireless Test Framework on Sasken Finland Oy:ssä kehitetty testausympäristö.

1 JOHDANTO

Kosketusnäytöt ovat viime aikoina yleistyneet huikeaa vauhtia matkapuhelinmarkkinoilla. Suurimman huomion kosketusnäyttöpuhelimista on saanut Applen julkaisema iPhone, vaikka esimerkiksi Sony Ericsson on tehnyt ja myynyt kosketusnäyttöä hyödyntäviä puhelimia jo useita vuosia. Myös Googlen kehittämä Android-käyttöjärjestelmä hyödyntää kosketusnäyttöjä.

Tämän työn tarkoitus on perehtyä Symbian-käyttöjärjestelmään perustuviin UIQ- ja S60-alustoihin ja kosketusnäytön hyödyntämiseen niissä. Työn teoriaosuudessa perehdytään Symbianin sovelluskehykseen, sen tukeen kosketusnäytölle ja siihen perustuviin matkapuhelinalustoihin. Työn ohjelmointivaiheessa toteutetaan testikehyksen siirtäminen UIQ-alustalle keskittyen toimivaan käyttöliittymään, joka hyödyntää kosketusnäyttöä.

2 SYMBIAN

Tässä luvussa tutustutaan Symbian-sovellusten rakenteeseen yleisesti, sekä annetaan yleiskuvaus kahdesta suurimmasta Symbianiin perustuvasta matkapuhelinalustasta. Lisäksi tutustutaan Symbianin tukeen kosketusnäytölle.

2.1 Symbianin sovelluskehys

Käyttäjäystävällisten matkapuhelinsovellusten tärkein ominaisuus on toimiva, graafinen käyttöliittymä. Symbian-alustalla graafisten sovellusten tukemisessa keskeisessä asemassa on sovelluskehyksen alijärjestelmä UIKON.

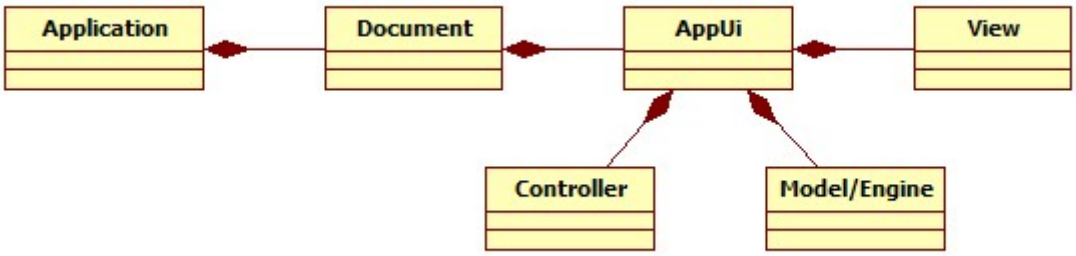
UIKON tarjoaa kehyksen, jonka avulla voidaan luoda graafisen käyttöliittymän sisältäviä sovelluksia. Kehyksen arkkitehtuuri kannustaa käyttämään MVC-mallia, jossa sovellus jaetaan erillisiin loogisiin osiin, joilla on sovelluksessa tietty tehtävä. MVC-mallin mukaiset tehtävät ovat malli (engl. model), näkymä (engl. view) sekä kontrolleri (engl. controller). UIKON tarjoaa perusluokat graafisten Symbian

sovellusten luontiin. Nämä luokat noudattavat MVC-mallia ja ovat listattuna taulukossa 1. /5, s. 312/

Luokka	**Kuvaus**
Application - CEikApplication	Sovelluksen käynnistyspiste, joka luo document-luokan ilmentymän, määrittää sovelluksen ominaisuuksia sekä tarjoaa rajapinnan sovelluksen resurssitiedostoon.
Document - CEikDocument	Toimii MVC-mallissa mallina. Sovelluksen ollessa tiedostopohjainen, document-olio on vastuussa tiedon tallentamisesta ja lataamisesta. Tämän luokan ilmentymä luo Application UI-luokan.
Application UI - CEikAppUi	Toimii MVC-mallissa kontrollerina. Vaikka tämä luokka on nimeltään Application UI, se on täysin näkymätön käyttäjälle. Tämän luokan ilmentymä luo näkymäluokan tai -luokkia sekä käsittelee usein käyttäjän komentoja, kuten valikoista tehtyjä valintoja.
View – luokka, joka periytyy CCoeControl:sta	Toimii MVC-mallissa näkymänä. Tämän luokan tarkoitus on näyttää sovelluksen tila käyttäjälle ja antaa käyttäjän kommunikoida sovelluksen kanssa. Myös näkymäluokka voi käsitellä käyttäjän komentoja, esimerkiksi näppäinpainalluksia.

Taulukko 1 *UIKON-kehyksen perusluokat graafisen sovelluksen luontiin* /5, s. 312/

Kaikkien graafisten Symbian-sovellusten tulee sisältää taulukossa 1 esiteltyjen luokkien toteutukset. Vaikka UIKONin tarjoamat luokat sinällään noudattavat MVC-mallia, käytetään Symbian-sovelluksissa usein kuitenkin erillisiä luokkia mallin toteuttamiseen. Kuvassa 1 on esitetty tyypillisen sovelluksen yleisen tason luokkakaavio.



Kuva 1 *MVC-mallisten sovellusten yleisluokkakaavio*

MVC-arkkitehtuurissa näkymä, kontrolleri ja malli ovat erillisiä komponentteja, joiden välillä kommunikointi tapahtuu tiettyjen sääntöjen mukaan. Komponentit ovat erillisiä yksiköitä, jolloin jonkun yksikön toteutusta voidaan muuttaa, ilman että se vaikuttaa toisiin yksiköihin.

2.2 Symbian-käyttöjärjestelmään perustuvat matkapuhelinalustat

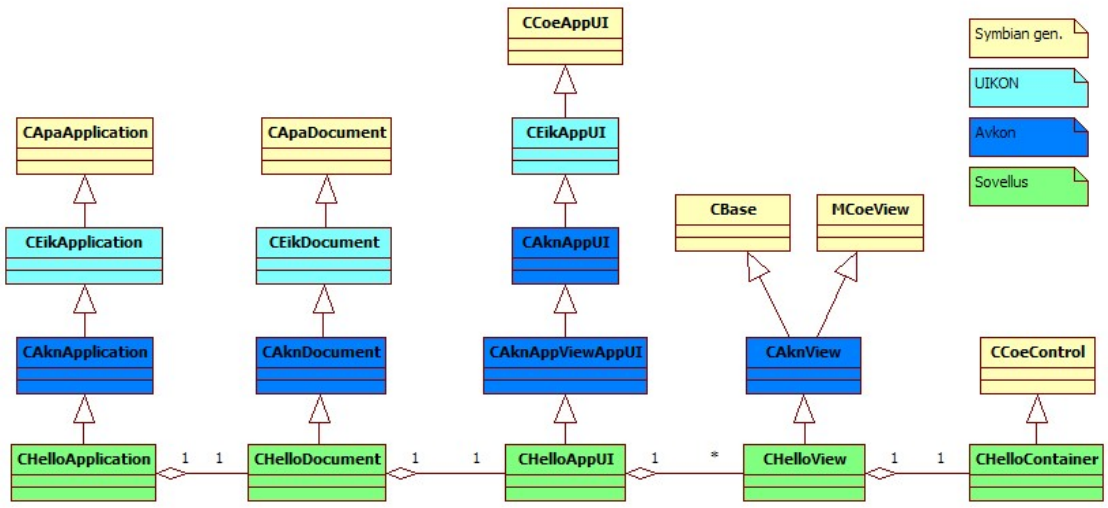
Symbian on matkapuhelimissa käytettävä käyttöjärjestelmä, johon perustuen on tehty useita eri matkapuhelinalustoja. Euroopassa tunnetuimpia alustoja ovat Nokian kehittämä ja käyttämä S60 ja esimerkiksi Sony Ericssonin ja Motorolan käyttämä UIQ. Kolmas tunnettu Symbianiin perustuva alusta on MOAP. Tätä alustaa käyttää NTT DOCOMO ja se on Aasiassa tunnetumpi kuin UIQ. Tässä opinnäytetyössä keskitytään S60- ja UIQ-alustoihin, eikä MOAP-alustaa käsitellä.

S60- ja UIQ-alustat perustuvat Symbianiin ja käyttävät täten hyväkseen kappaleessa 2.1 esitettyä UIKON-kehystä ja sen luokkia. Eri alustat laajentavat Symbianin tarjoamaa kehystä omilla kirjastoillaan, joiden avulla käyttöliittymä saadaan muokattua kohdelaitteille sopivaksi. Kohdelaitteissa on eroja esimerkiksi näytön resoluution ja orientaation sekä eri vuorovaikutustapojen suhteen. Laitetta voidaan käyttää esimerkiksi erilaisilla näppäimistöillä tai kosketusnäytön avulla. /5, s. 313/

Vaikka eri alustat on suunniteltu hieman eri tavoilla ja erilaisin tavoittein, on pohjalla kuitenkin yhteinen käyttöjärjestelmä. Yhteisen käyttöjärjestelmän vuoksi, alustojen ohjelmointirajapinnat ovat usein saman kaltaisia. S60-alustan lisäämän

kirjaston nimi on Avkon ja UIQ:n Qikon. Käytännössä suurin osa näiden alustojen luokista periytyy Symbianin luokista, ja niihin on vain lisätty eri alustoilla tarvittavia ominaisuuksia. /5, s. 313/

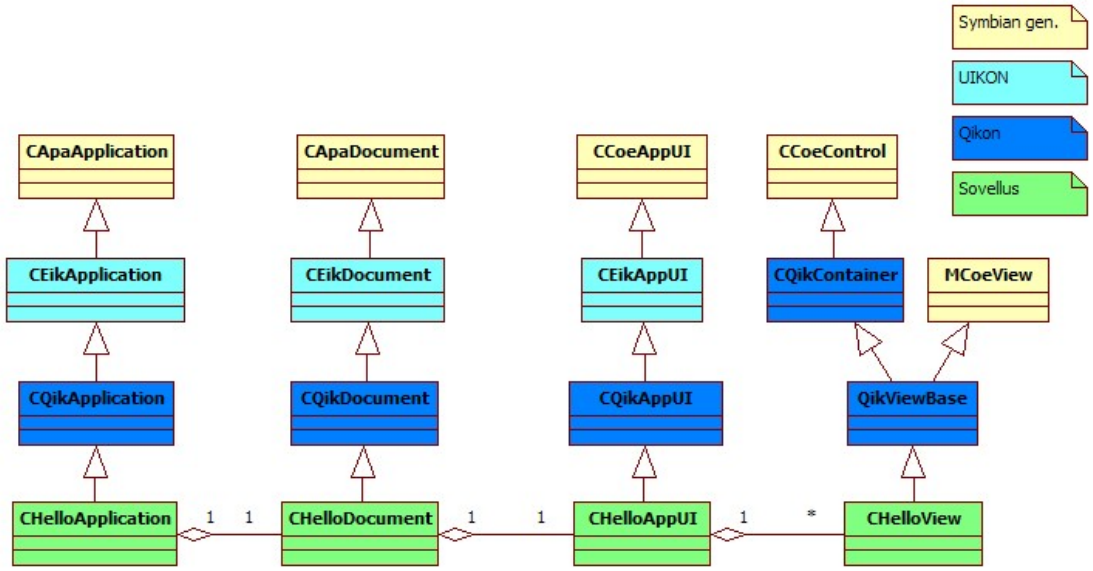
Koska molemmat alustat perustuvat samaan käyttöjärjestelmään, on S60- ja UIQ-sovellusten rakenne lähes sama. Vaikka rakenne on saman kaltainen, saadaan Avkon- ja Qikon-alustojen avulla sovelluksista ja käyttöliittymistä erilaiset. Molemmat alustat tukevat useita erilaisia käyttöliittymiä, kuten erilaisia näytön resoluutioita ja orientaatioita. S60-sovelluksen rakenne sekä luokkien periytyminen on esitetty kuvassa 2.



Kuva 2 *S60-sovelluksen luokkien periytyminen*

Yllä olevasta kuvasta nähdään, että Avkon-alustan luokat voidaan helposti tunnistaa Akn-etuliitteestä. Vastaavasti UIKON-kehyksen luokkien alussa käytetään etuliitettä Eik. Eik-etuliite johtuu siitä, että alunperin Symbian-käyttöjärjestelmän käyttöliittymän sovelluskehyksen nimi oli Eikon ja käyttöjärjestelmä itse tunnettiin nimellä EPOC. Myöhemmin Eikon-kehyksen nimi muutettiin UIKONiksi, mutta luokkien nimiä ei vaihdettu. Jos luokan nimessä ei ole Akn- eikä Eik-etuliitettä, on kyseessä geneerinen Symbian-luokka. Tämä ei koske kuvan alinta riviä, jonka luokat sovelluksen tekijä voi nimetä kuten haluaa. /5, s. 311-313/

Kuvasta nähdään myös, että S60-alustalla näkymäluokka ei periydy CCoeControl-luokasta, eli se ei itse ole kontrolli. Tämä on poikkeavaa UIQ:hun verrattuna, sillä UIQ:ssa näkymä periytyy suoraan CCoeControlista, joten se on itsekin kontrolli. Tämän ansiosta UIQ-sovelluksissa ei ole erillistä container-luokkaa. UIQ-sovelluksen rakenne sekä luokkien periytyminen on esitetty kuvassa 3.



Kuva 3 *UIQ- sovelluksen luokkien periytyminen*

Kuten S60:n, myös UIQ:n luokat on nimetty tiettyä etuliitettä käyttäen. UIQ-alustalla etuliite on Qik. Myöhemmissä kappaleissa tutustutaan hieman tarkemmin näihin alustoihin sekä erityisesti niiden käyttöliittymien tyyliin ja käyttölogiikkaan.

2.3 Symbianin tuki kosketusnäytölle

Eräs tämän opinnäytetyön lähtökohdista oli tutustua kosketusnäyttöä hyödyntävän sovelluksen tekoon UIQ-alustalla. Kun UIQ-alustaa kehitettiin, oli alusta asti selvä, että sen piti tukea kosketusnäyttöä. Kosketusnäytön hyödyntämisen mahdollistaa Symbianin tuki niille. Kosketusnäytön koskettaminen saa aikaan kosketustapahtuman (engl. pointer event), joka on osittain samanlainen kuin puhelimen fyysiseltä näppäimistöltä tuleva tapahtuma.

Kosketus- ja näppäintapahtumien käsittely poikkeaa selvästi toisistaan. Jotta näemme kosketusnäytön kautta tulevien tapahtumien käsittelyn erot perinteiseltä näppäimistöltä tulevien tapahtumien käsittelyyn, tutustumme seuraavissa kappaleissa molempiin tapahtumiin.

2.3.1 Näppäintapahtumien käsittely

Näppäintapahtuma voi tulla joko näppäimen painalluksesta tai front-end prosessorilta (FEP), joka on eräänlainen esiprosessori. FEP:tä käytetään esimerkiksi ennakoivan tekstinsyötön, käsialan tunnistuksen, äänen tunnistuksen sekä kosketusnäytöllä olevan virtuaalinäppäimistön tapahtumien käsittelyyn. Esiprosessori tuottaa näppäintapahtumia esimerkiksi virtuaalinäppäimistön tietyn näppäimen painalluksesta. Kun näppäintä painetaan, FEP käsittelee tapahtuman ja muuntaa sen sopivaksi näppäintapahtumaksi. Sovelluksen ei tarvitse tietää, tuleeko näppäintapahtuma fyysiseltä näppäimistöltä vai FEP:ltä, koska tapahtuman käsittely on molemmissa tapauksissa sama. /9; 5, s. 432-433/

Näppäintapahtumat käsitellään kontrollipinon (engl. control stack) OfferKeyEventL-funktiossa, joka kutsuu vastaavaa funktiota jokaiselle pinossa olevalle kontrollille. Niitä selataan, kunnes joku niistä käsittelee tapahtuman ja palauttaa tätä vastaavan arvon tai kaikki kontrollit on käyty läpi, eikä mikään kontrolli ole käsitellyt tapahtumaa. Jos mikään kontrolli ei käsitellyt tapahtumaa, tulee se konrollipinon omistajan HandleKeyEventL-funktioon käsiteltäväksi. /5, s. 434/

Sovelluksen oman logiikan pitää hoitaa näppäintapahtumien käsittely. Esimerkiksi yhdistelmäkontrollit (engl. compound control) voivat itse käsitellä tapahtuman, tarjota tapahtumaa jollekin osakontrollille käsiteltäväksi tai olla käsittelemättä sitä ollenkaan. Sovelluksen sisäinen toteutus siis määrää, mikä kontrolli tapahtuman käsittelee ja mihin se vaikuttaa. Yksi esimerkki näppäintapahtumien käsittelystä voisi olla sellainen, jossa yhdistelmäkontrolli käsittelee nuolinäppäimet ja vaihtaa niiden avulla osakontrolliensa fokusta. Osakontrollit käsittelisivät muita näppäimiä tarpeensa mukaan. /5, s. 432, s. 434-437/

2.3.2 Kosketustapahtumien käsittely

Tässä kappaleessa tutustutaan kosketustapahtumien kulkuun järjestelmässä, niiden käsittelyyn, niiden eroihin näppäintapahtumiin verrattuna ja erilaisiin kosketustapahtumatyyppeihin.

2.3.2.1 Kosketustapahtuman kulku ja käsittely yleisesti

Kosketustapahtumat eroavat näppäintapahtumista siten, että ne kohdistuvat suoraan johonkin tiettyyn kontrolliin, eikä sovelluksen oman logiikan tarvitse hoitaa niiden välittämistä halutulle kontrollille. Ikkunapalvelin (engl. window server) assosioi tapahtuman tiettyyn, esillä olevaan ikkunaan, jonka alueella kosketus tapahtui, ja välittää tapahtuman ikkunaryhmän omistamalle sovellukselle. /5, s. 440-441/

Sovelluksen sisällä kosketustapahtumien käsittely etenee seuraavasti: Ensin tapahtuma välitetään Application UI-luokalle, jonka HandleWsEventL-funktiossa se tunnistetaan tiettyyn ikkunaan assosioiduksi kosketustapahtumaksi. Funktiosta tehdään kutsu ikkunan omistaman kontrollin ProcessPointerEventL-funktioon, josta kutsutaan HandlePointerEventL-funktiota. Tässä funktiossa voidaan tehdä halutut asiat kosketustapahtuman sattuessa. Jos kontrolli, jolle tapahtuma saapui, on yhdistelmäkontrolli, on sen oletustoteutus sellainen, että se käy läpi kaikki näkyvät osakontrollinsa ja välittää tapahtuman sille kontrollille, joka sijaitsee sillä kohtaa, jossa tapahtuma tapahtui. /5, s. 440-441/

Kosketustapahtumien käsittely on siinä mielessä samanlaista kuin näppäintapahtumien, että kontrolli voi olla käsittelemättä tapahtumaa tai se voi käsitellä tapahtuman ja tuottaa esimerkiksi komennon, joka käsitellään muualla. Toisaalta, kun kosketustapahtumat toimitetaan suoraan tietylle kontrollille, ei kosketustapahtumia normaalisti tarvitse välittää toisille kontrolleille, eikä tapahtumaa tarjota järjestelmän toimestakaan muille kontrolleille. /5, s. 441/

2.3.2.2 Eri kosketustapahtumatyypit ja niiden käsittely

Kosketustapahtumaa edustaa TPointerEvent-tyypin ilmentymä, joka on samankaltainen kuin näppäintapahtumaa edustava TKeyEvent. TPointerEvent sisältää tiedon siitä, oliko tapahtuman tyyppi kynän alas painaminen vai ylös nostaminen. Lisäksi saadaan tieto esimerkiksi tapahtuman toistosta ja sijainnista sekä siitä, onko Shift-, Ctrl- tai Alt-näppäin painettuna. Tapahtuman toistolla tarkoitetaan sitä, kun kynää pidetään painettuna samassa kohdassa pidemmän aikaa. /5, s. 439-440/

Kosketustapahtumien käsittelyssä pitää ottaa huomioon tapahtuman tyypin ja sijainnin lisäksi muitakin asioita. Tapahtumien käsittelyssä on huomioitava, että kynää voidaan siirtää näytön pinnalla, jolloin kynän nostamistapahtuma ei välttämättä tapahdu saman kontrollin kohdalla, kuin kynän laskemistapahtuma. Mahdollinen on myös tilanne, jossa ei haluta antaa kosketustapahtumia millekään muulle, kuin jollekin tietylle kontrollille. Näiden tilanteiden hallitsemiseksi Symbian tarjoaa seuraavat tekniikat: pointer grab ja pointer capture. /5, s. 440/

Pointer grab on tekniikka, jonka avulla varmistetaan, että kontrolli, jonka päällä tapahtuu kynän alas painaminen, saa käsiteltäväkseen myös seuraavat kosketustapahtumat, kunnes tulee seuraava kynän nostotapahtuma. Kynän nostotapahtuma lähetetään siis samalle kontrollille, vaikka kynä nostettaisiinkin ylös jonkun muun kontrollin kohdalla. Esimerkiksi näytöllä näkyvää nappia painettaessa piirretään nappi uudelleen, siten, että käyttäjä näkee sen olevan painettuna. Kun käyttäjä nostaa kynän näytön pinnalta, piirretään näppäin uudelleen ylääsentoon, ja jos kynä oli nostotapahtuman aikana saman näppäimen päällä, suoritetaan siihen liitetyt toiminnot. /5, s. 572-573/

Pointer grabin toimimiseksi pitää ikkunapalvelimen muistaa oikea ikkuna ja kontrolliympäristön pitää muistaa oikea kontrolli. Tämän ominaisuuden toteuttaa aiemmin mainittu ProcessPointerEventL-funktio, joka tekee myös muuta hyödyllistä esikäsittelyä kosketustapahtumille, kuten

- kynällä painaminen-tapahtuman hylkääminen näkymättömiltä kontrolleilta
- tapahtumien hylkääminen, kunnes tulee seuraava kynän nostamistapahtuma

- himmennettyjen kontrollien kosketustapahtumien raportointi ja hylkääminen
- kynällä painaminen-tapahtumien raportointi sellaisten kontrollien kohdalla, jotka eivät ole fokusoituja, mutta jotka ovat kykeneviä ottamaan fokuksen /5, s. 573, s. 441-442/

Pointer capture on tekniikka, jonka avulla voidaan estää kaikkia muita kontrolleja saamasta käsiteltäväkseen kynällä painaminen-tapahtumia. Tätä tekniikkaa hyödynnetään esimerkiksi dialogeissa, jolloin voidaan unohtaa kaikki dialogin ulkopuolella tapahtuneet kosketukset, sekä valikoissa, jolloin kosketus valikon ulkopuolella voidaan käsittää haluksi sulkea valikko. Kun valikko saa käsiteltäväkseen myös sen ulkopuolella tapahtuneen kosketuksen, se voi sulkea itsensä. /5, s. 440/

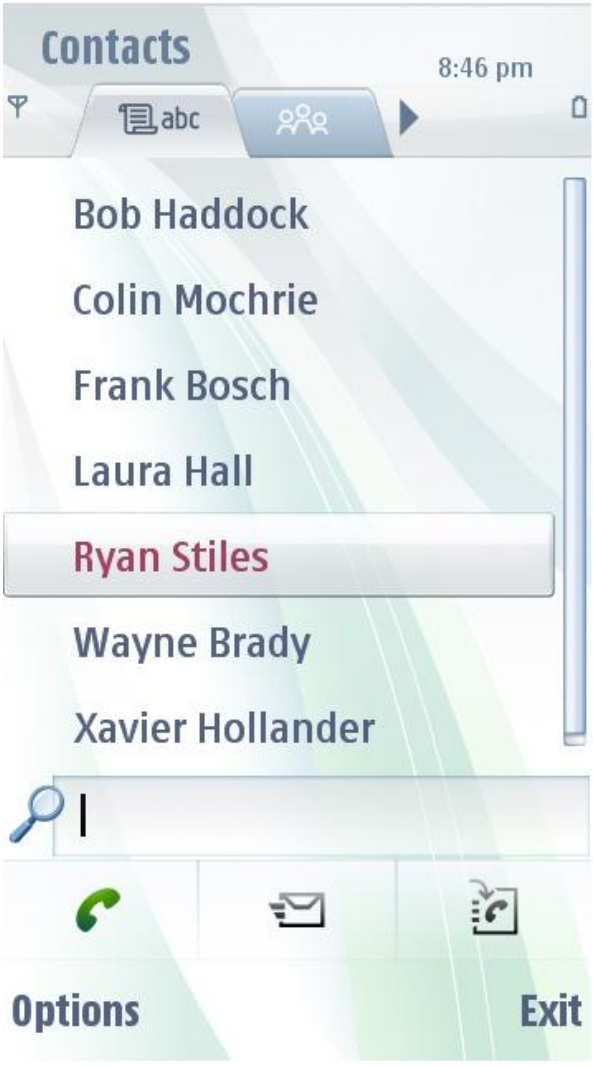
3 S60-ALUSTA

S60-alustan uusin versio, 5.0, julkistettiin syksyllä 2008. Uusin versio perustuu Symbianin versioon 9.4, joka toi ensimmäistä kertaa tuen kosketusnäytölle S60-alustalle. Tässä kappaleessa tutustutaan hieman tarkemmin S60-alustan sovellusten käyttöliittymään sekä alustan tukeen kosketusnäytölle.

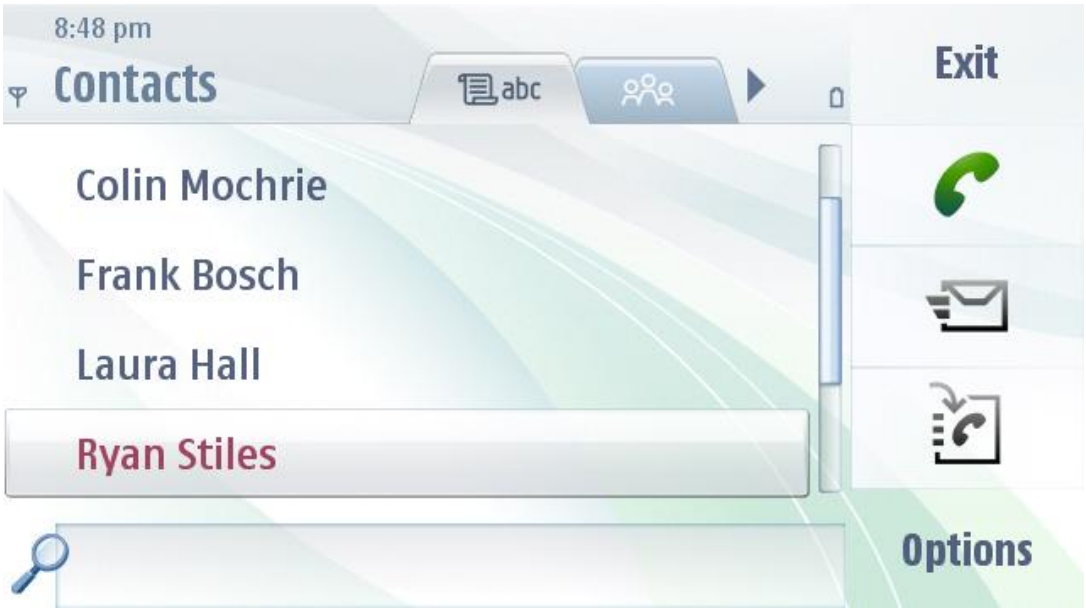
3.1 S60-sovellusten tyyli ja käyttölogiikka

S60-alusta on suunniteltu siten, että puhelimia on mahdollista käyttää yhdellä kädellä. Puhelimissa on perinteisesti ollut fyysinen näppäimistö ja näytöllä näkyvä käyttöliittymä on perustunut näytön alalaidassa olevien toimintonäppäinten (engl. Softkey) käyttöön. Sovelluksen tekijä voi määritellä, mitä toimintonäppäimet tekevät, mutta hyvin yleisesti vasemman puoleinen toimintonäppäin avaa valikon ja oikeanpuoleisella poistutaan sovelluksesta tai siirrytään edelliseen näkymään. Kun sovelluksesta poistutaan oikeaa toimintonäppäintä tai valikkoa käyttäen, sovellus suljetaan, eikä sitä jätetä taustalle. Jos sovellus halutaan jättää taustalle, jolloin sovelluksen käyttöä voi jatkaa myöhemmin samasta tilasta, pitää sovelluksesta poistua erityistä valikkonäppäintä käyttäen.

S60-alustan 5.0-versio vaikuttaa perusidealtaan hyvin samanlaiselta aiempiin versioihin verrattuna. Käyttöliittymän tyyli hyödyntää edelleen toimintonäppäimiä ja sovellusten käyttö onnistuu edelleen yhdellä kädellä, sormia käyttäen, vaikka puhelimessa olisikin kosketusnäyttö. Alla olevissa kuvissa 4 ja 5, on kuvakaappaukset S60:n 5.0-emulaattorin yhteystietosovelluksesta.



Kuva 4 *S60:n 5.0-emulaattorin yhteystietonäkymä, pystysuuntainen näyttö*



Kuva 5 *S60:n 5.0-emulaattorin yhteystietonäkymä, vaakasuuntainen näyttö*

Yllä olevissa kuvissa on toimintonäppäimien lisäksi nähtävissä muita näppäimiä näytön oikeassa ja alareunassa. Ne kuuluvat uuteen Toolbar APIin, jonka avulla sovellukseen saadaan lisättyä helposti ja nopeasti kosketettavissa olevia näppäimiä, joihin voidaan sijoittaa tärkeitä toimintoja. Uudet näppäimet parantavat käyttöliittymää huomattavasti, kun toimintoja ei tarvitse sijoittaa esimerkiksi valikkoon, josta niiden valitseminen olisi nykyistä selvästi vaivalloisempaa.

3.2 Kosketusnäyttö S60-alustalla

Tässä kappaleessa tutustutaan hieman tarkemmin kosketusnäyttöön S60-alustalla ja kerrotaan alustan 5.0-version mukana tulleista uusista ohjelmointirajapinnoista, jotka mahdollistavat paremman käyttöliittymän luomisen.

3.2.1 Yleistä kosketusnäytöstä S60-alustalla

Kosketusnäytön tuominen S60-alustalle on tuonut mukanaan monia uudistuksia. Eräs helpoiten havaittavista uudistuksista on tuki uudelle resoluutiolle, jonka myötä myös laitteiden näytön koko kasvaa. Alustan 5.0-versio tukee vanhan 240x320-resoluution lisäksi 640x360-resoluution omaavia kosketusnäyttöjä. /6/

Kosketusnäyttöä voi ohjata joko sormella tai kynällä ja se antaa käyttäjälle palautetta tactile feedback ominaisuuden avulla. Tactile feedback tarkoittaa sitä, että näytöllä näkyvät UI-komponentit voivat rekisteröidä tietyt alueet antamaan käyttäjälle palautetta uuden API:n avulla. Kun käyttäjä koskettaa esimerkiksi nappia, joka on rekisteröinyt itsensä antamaan palautetta, annetaan puhelimen värinätoiminnon avulla käyttäjälle tieto siitä, että painallus on tullut perille. /6/

Kosketusnäyttöä ajatellen, on S60:n 5.0-version käyttöliittymään tehty parannuksia ja lisäyksiä. Yksi lisäys on esimerkiksi uusi toolbar-komponentti, joka toimii työkalupalkkina, jonka avulla voidaan suorittaa tärkeitä toimintoja suoraan näytöllä näkyvää kohdetta koskettamalla. Myös tekstinsyöttö on uudistunut. Mahdollisuuksia on nyt perinteisten fyysisten numero- ja qwerty-näppäimistöjen lisäksi pieni qwerty-virtuaalinäppäimistö, käsialan tunnistukseen pohjautuva kirjoitus sekä koko näytön qwerty-näppäimistö. /6/

3.2.2 S60:n 5.0-versiossa esitellyt käyttöliittymärajapinnat

S60:n 5.0-versiossa julkaistiin useita uusia ohjelmointirajapintoja. Osa uusista rajapinnoista tuo alustaan uusia käyttöliittymäkomponentteja, jotka on suunniteltu kosketusnäyttöä ajatellen. Uudet käyttöliittymärajapinnat on esitetty taulukossa 2.

Ohjelmointirajapinta	**Kuvaus**
Choise List API	Mahdollistaa valintojen tekemisen pystysuuntaisesta listasta.
Generic Button API	Mahdollistaa sovellusten piirtää mielivaltaisen napin, jossa on ikoni tai teksti. Napin ominaisuudet voidaan määritellä, kuten onko se näkyvissä tai painettuna.
Hierarchical Lists API	Mahdollistaa datan esittämisen hierarkkisessa listassa käyttäen tekstiä ja grafiikkaa. Rajapinta tarjoaa kolme listatyyppiä: -yleiskäyttöinen hierarkkinen treelist -yhden sarakkeen lista, jossa alkio voidaan laajentaa ilman sisennystä -single-style tree, jolla kuvataan kansio- ja tiedostorakennetta. Tämä API julkaistiin jo S60 3rd Edition, Feature Pack 2 SDK:ssa, mutta toteutusta ei ole ollut missään laitteessa ennen S60:n 5.0-versiota.
Stylus Pop-up Menu API	Mahdollistaa sovellusten piirtää pop-up valikon, jossa on lista valintoja. Lista luodaan kohtaan, jossa kosketus näyttöön havaitaan.
Tactile Feedback Client API	Mahdollistaa sovellusten ja käyttöliittymä komponenttien rekisteröidä tactile feedback alueensa ja vastaanottaa tietoja kosketustapahtumista.
Title Pane Touch Observer API	Mahdollistaa sovellusten seurata käyttäjän painalluksia näytön Title Pane-alueelle.
Toolbar API	Mahdollistaa sovellusten rakentaa kiinteän tai liikkuvan työkalupalkin sovelluksen näkymään. Työkalupalkki koostuu AVKON-napeista, joita se voi sisältää korkeintaan neljä kappaletta.
Touch UI Utilities API	Tällä hetkellä tarjoaa vain palvelun, jolla tunnistetaan pitkä painallus kosketusnäytöllä.

Taulukko 2 *S60:n 5.0-version uudet käyttöliittymän ohjelmointirajapinnat* /2/

4 UIQ-ALUSTA

UIQ on suunniteltu siten, että laitevalmistajat voivat tehdä hyvin erilaisia puhelimia ja sovellukset toimivat niissä kaikissa. Kun sama koodi toimii kaikissa puhelimissa, on laitevalmistajien helppo tehdä erilaisia puhelimia. Myös kolmansien osapuolten työ helpottuu huomattavasti, kun sama koodi toimii kaikissa puhelimissa. Laitteet voivat erota toisistaan esimerkiksi näytön sekä näppäimistön osalta. Kuten kappaleessa 2.3 mainittiin, on UIQ tukenut kosketusnäyttöjä ensimmäisestä versiostaan alkaen. Seuraavissa kappaleissa perehdytään alustan 3.0-versioon. /8/

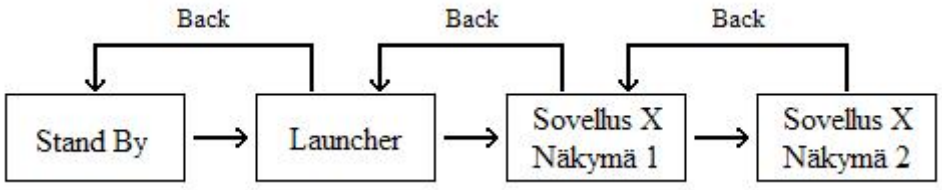
4.1 UIQ-sovellusten tyyli ja käyttölogiikka

Tässä kappaleessa tutustutaan UIQ-sovellusten yleiseen tyyliin sekä niiden toimintaan. Lisäksi kuvataan lyhyesti sovellusten rakennetta.

4.1.1 Yleistä

UIQ-sovellukset on tarkoitettu jäämään taustalle, kun niitä ei käytetä. Tällöin käyttäjä voi jatkaa sovelluksen käyttöä samasta tilanteesta, johon hän sen jätti poistuessaan siitä. Käyttäjälle ei anneta mahdollisuutta sulkea sovellusta, paitsi erillistä sovellusten hallintatyökalua käyttäen. Käytännössä tämä tarkoittaa sitä, että sovelluksissa ei ole valikossa eikä toimintonäppäimissä sovelluksen sulkemisvaihtoehtoa. /3/

Sovelluksesta poistuminen sekä näkymien välillä navigointi UIQ-alustalla tapahtuu erityisellä back-näppäimellä. Aina kun uusi näkymäluokka luodaan, kutsutaan sen kantaluokan CQikViewBasen rakentajaa antamalla sille parametrina toisen näkymän id, jonka halutaan toimivan uuden näkymän vanhempana. Kun jokaiselle näkymälle määritellään vanhempana toimiva näkymä, voidaan näkymien välillä siirtyä back-näppäintä painettaessa. Esimerkki sovelluksen navigointijärjestyksestä on nähtävissä kuvassa 6. /12/



Kuva 6 *Näkymien välillä navigointi UIQ-alustalla*

Kuvassa 6 on kuvattu siirtyminen puhelimen stand by-näkymästä launcherin kautta sovellukseen Sovellus X. Kun sovelluksen Näkymä 1 luodaan, annetaan kantaluokan rakentajalle parametrina arvo KNullViewId, joka tarkoittaa, että näkymällä ei ole samassa sovelluksessa vanhempaa, vaan näkymästä siirrytään sovelluksen ulkopuolelle back-näppäintä painettaessa. Kun luodaan toinen näkymä, annetaan parametrina edellistä näkymää vastaava id, jolloin se näkymä rekisteröityy toisen näkymän vanhemmaksi. Normaalisti back-näppäin välitetään kantaluokalle käsiteltäväksi. Sovelluskehys käsittelee komennon, tarkastamalla, mikä näkymä on asetettu nykyisen näkymän vanhemmaksi ja vaihtamalla näkymän siihen. /12/

Koska kaikki sovellukset jäävät oletuksena taustalle, kuluu muistia enemmän, kuin jos sovelluksia suljettaisiin. Jos puhelimen muisti käy vähiin, voi UIQ-alustan muistia hallitseva Memory Manager kehys pyytää sovellusta sulkeutumaan. Oletuksena kehys kutsuu document-luokan SaveL-funktiota, jossa sovelluksen tulee tallentaa kaikki tallentamattomat tietonsa. /3/

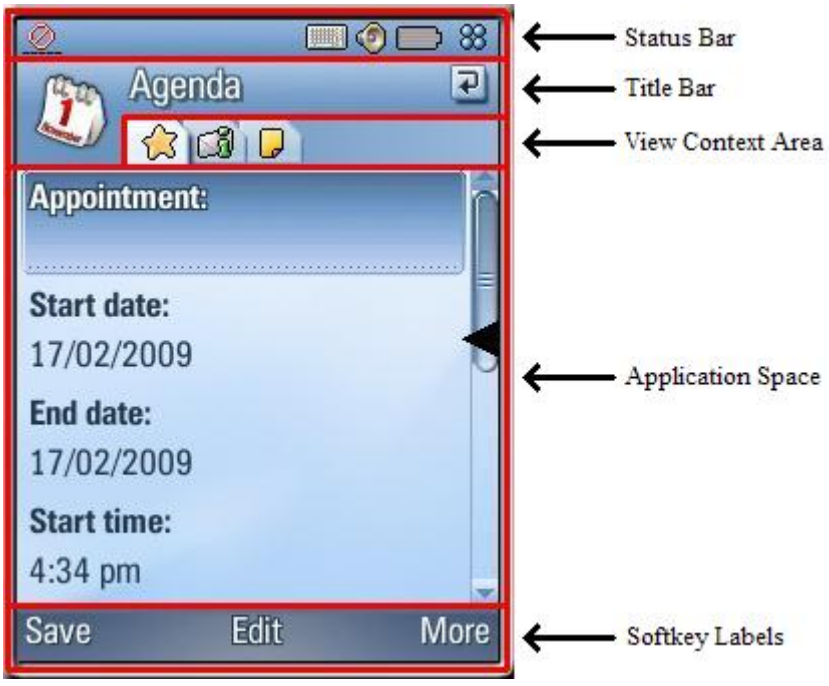
4.1.2 UIQ:n eri UI-konfiguraatiot

UIQ:n tuki erilaisille laitteille muodostuu UI-konfiguraatioista. Jokainen konfiguraatio muodostuu joukosta parametreja. Niin sanottuja käyttöliittymäparametreja ovat näytön resoluutio ja orientaatio, näytön kosketustuki sekä käyttöliittymän tyyli. UIQ:n 3.0-versio tukee kahta eri näytön resoluutiota: 240x320, sekä 240x256 pikseliä. Molemman kokoiset näytöt voivat olla käännettynä myös vaakatasoon, molempiin suuntiin. /8/

Käyttöliittymän tyyli voi UIQ-alustalla olla joko kynä- tai toimintonäppäintyyli. Kynätyyli on optimoitu kynän käyttöä varten, mutta sitä voidaan käyttää myös ilman kynää. Toimintonäppäintyyli on perinteinen, myös S60-alustan käyttämä tyyli, joka on optimoitu yhden käden käyttöä varten. Toimintonäppäintyyliä voidaan käyttää kosketusnäytöllä tai ilman, jolloin käyttöliittymä on hieman erilainen. Käytännössä tämä tyyli on hyvin lähellä S60-alustan tyyliä, mutta joitakin erojakin on. Seuraavissa kappaleissa perehdytään näihin kahteen tyyliin. /8/

4.1.2.1 Toimintonäppäintyylinen käyttöliittymä

Toimintonäppäintyylinen näkymä näyttöalueineen on nähtävissä kuvassa 7.



Kuva 7 *Toimintonäppäintyyli, tuki kosketusnäytölle*

Kuvassa 7 on ylimpänä Status Bar, joka sisältää tietoja järjestelmän tilasta, kuten akun varauksesta ja verkkoyhteyden tilasta. Tämän alueen sisältö määräytyy laitevalmistajan määrityksistä, eikä sovelluksen kehittäjä voi vaikuttaa siihen. Yllä olevassa kuvassa on kosketusnäyttöä tukeva tyyli, jossa Status Barissa näkyviä ikoneja voidaan koskettaa ja saada näin tarkempaa tietoa esimerkiksi akun varauksesta. /10/

Status Barin alapuolella on Title Bar eli otsikkopalkki. Tämä alue sisältää yleensä ainakin sovelluksen nimen tai muun tekstin sekä sovelluksen ikonin. Kuvassa näkyy, että kosketusnäyttöä hyödyntävässä toimintonäppäintyylissä Title Barissa on paluunappi, jota käytetään kappaleessa 4.1.1. kuvatun mukaiseen navigointiin näkymien välillä. /10/

View Context Area on alue, jolla sovellus voi näyttää monenlaista tietoa tai esimerkiksi helpottaa navigointia näkymien välillä välilehtien avulla. Alueelle voidaan lisätä myös esimerkiksi ikoneita tai jonkin toiminnon etenemistä kuvaava palkki. /10/

Application Space on alue, jolle sovelluksen näkymät piirretään. Tämä alue ei poikkea juurikaan S60-alustan versiosta. Ainoa ero tässä alueessa eri konfiguraatioiden välillä on vierityspalkin leveys, joka on kosketusnäyttöä käytettäessä hieman suurempi. Kuten S60-alustalla, voidaan UIQ-alustallakin ottaa koko näyttö käyttöön sovellusta varten, mutta silloin pitää olla hyvin tarkkana käyttöliittymän käyttäjäystävällisyyden kanssa. /10/

Alimmaisena toimintonäppäintyylisessä konfiguraatiossa näytöllä on Softkey Labels alue, joka sisältää käytännössä vain otsikot, jotka kuvaavat, mitä tapahtuu, kun nappia painetaan. Toimintonäppäimet toimivat UIQ-alustalla samaan tapaan kuin S60-alustallakin, mutta näppäinten järjestys on erilainen. UIQ-alustalla valikko avataan oikeanpuoleista näppäintä painamalla. /10/

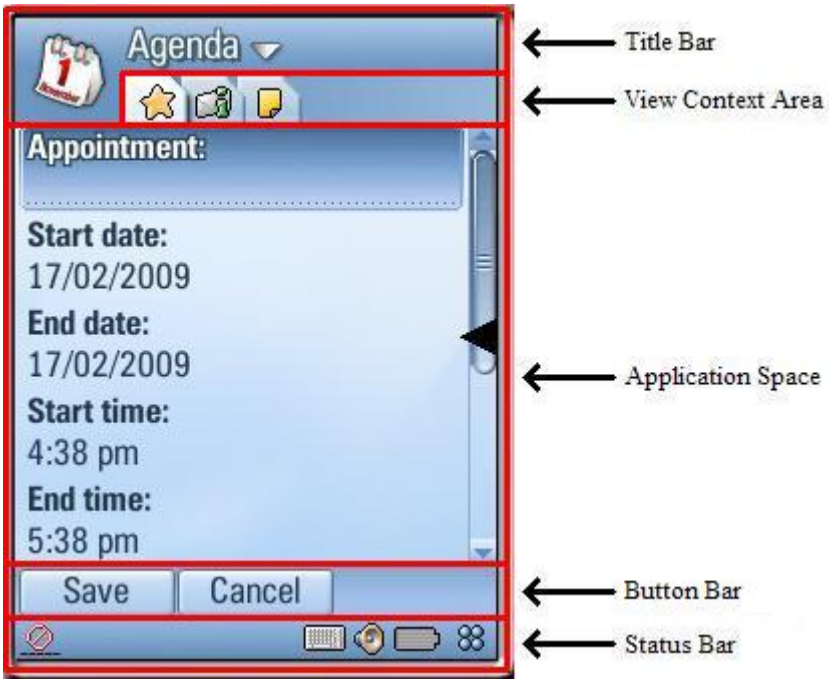
Kuten aiemmin mainittiin, voidaan toimintonäppäintyyliä käyttää myös ilman kosketusnäyttöä. Kuvassa 8 on esitetty näkymä toimintonäppäintyylistä ilman kosketusnäyttöä.

Kuva 8 *Toimintonäppäintyyli, ei tukea kosketusnäytölle*

Kuten kuvasta 8 voidaan huomata, ovat näkymät hyvin samankaltaisia kosketusnäytöllä ja ilman. Voidaan sanoa, että kosketusnäyttö tuo tiettyjä lisämahdollisuuksia käyttöliittymään, jotka voivat helpottaa toimintojen tekemistä ja parantaa käyttäjäystävällisyyttä. Ilman kosketusnäyttöä, ei Status Barin kautta saada suoraan tarkempia tietoja järjestelmän tilasta. Myös paluunäppäin puuttuu ja on toteutettu tällaisessa laitteessa fyysisenä näppäimenä.

4.1.2.2 Kynätyylinen käyttöliittymä

Kynätyylin erot toimintonäppäintyyliin verrattuna on melko helppo havaita katsomalla kuvaa 9. Näytön alueet ovat pääosin samat, mutta niiden järjestys on osittain vaihtunut. Merkittävin ero tyylien välillä on se, että kynätyylissä toimintonäppäimet on korvattu näppäinpalkilla (Button Bar). Kyseisellä alueella on yleensä toimintonäppäintyylistäkin tuttu paluunappi sekä muita nappeja, joita sovelluksen tekijä sinne on halunnut sijoittaa. /10/



Kuva 9 *Kynätyyli, pystysuuntainen näyttö*

Title Barissa on kynätyylissä heti sovelluksen nimen jälkeen nuoli-ikoni, jota koskettamalla saadaan näkyviin valikko. Valikko on sama kuin toimintonäppäintyylissä oikeanpuoleista toimintonäppäintä painettaessa aukeava valikko. Jotta kosketusnäytön käyttö olisi helppoa ja nopeaa, on UIQ:ssa kuitenkin mahdollista määritellä valikon alkioille tietyllä lipulla, halutaanko niiden olevan mieluummin valikossa vai Button Barissa. Jos alkiolle on lipulla määritelty, että sen halutaan mieluummin olevan Button Barissa helposti ja nopeasti valittavissa, niin silloin Button Bariin tulee tätä komentoa vastaava nappi, eikä komentoa löydy Title Barista aukeavasta valikosta. /10/

Kuten aiemmin mainittiin, tukee UIQ myös näytön käyttöä vaakasuunnassa. Kuvassa 10 on esimerkki UIQ:n näkymästä, kun näyttöä käytetään vaakasuunnassa.

Kuva 10 *Kynätyyli, vaakasuuntainen näyttö*

Yllä olevassa kuvassa näkyy myös hyvin, että Button Barissa voidaan käyttää tekstin sijaan ikoneita näppäimissä, jolloin kyseiselle alueelle voidaan sijoittaa enemmän toimintoja. /10/

4.1.2.3 Eri UI-konfiguraatioihin varautuminen

UIQ-alusta tukee eri käyttöliittymäkonfiguraatioita ja näin ollen erilaisia puhelimia. Jotkut puhelimet tukevat useita eri UI-konfiguraatioita. Konfiguraatioiden vaihtaminen on joissain tapauksissa mahdollista tehdä sovelluksen sisältä mutta joissakin tapauksissa vaihto voi tapahtua vain jonkun järjestelmän tapahtuman yhteydessä. Sovellus voi esimerkiksi vaihtaa näytön orientaatiota mutta esimerkiksi joissakin puhelimissa oleva aukeava mekanismi, joka paljastaa fyysisen näppäimistön, saa aikaan konfiguraation vaihdon jota ei voi sovelluksesta ohjelmallisesti tehdä. /11/

Eri käyttöliittymäkonfiguraatioita varten voidaan sovelluksen resursseissa määritellä erilliset näkymät ja komennot, joita käytetään eri konfiguraatioilla. Normaalisti sovellusten ei tarvitse määritellä eri konfiguraatioille eri näkymiä ja komentoja. Joissakin erityistapauksissa voidaan kuitenkin haluta sovelluksen näyttävän erilaiselta tai sisältävän eri komentoja eri konfiguraatioilla. Jokainen sovelluksen näkymä voi määritellä, mitä käyttöliittymäkonfiguraatioita se tukee.

Määrittely tehdään resursseissa QIK_VIEW_CONFIGURATIONS-rakenteen avulla. Kuvassa 11 on esitetty, kuinka tämä määrittely tehdään. /11/

```
RESOURCE QIK_VIEW_CONFIGURATIONS r_ui_configurations
    {
    configurations =
        {
        QIK_VIEW_CONFIGURATION
            {
            ui_config_mode = KQikPenStyleTouchPortrait;
            command_list = r_Hello_commands;
            view = r_Hello_layout;
            },
        QIK_VIEW_CONFIGURATION
            {
            ui_config_mode = KQikPenStyleTouchLandscape;
            command_list = r_Hello_commands;
            view = r_Hello_layout;
            },
        QIK_VIEW_CONFIGURATION
            {
            ui_config_mode = KQikSoftkeyStyleTouchPortrait;
            command_list = r_Hello_commands;
            view = r_Hello_layout;
            },
        QIK_VIEW_CONFIGURATION
            {
            ui_config_mode = KQikSoftkeyStylePortrait;
            command_list = r_Hello_commands;
            view = r_Hello_layout;
            },
        QIK_VIEW_CONFIGURATION
            {
            ui_config_mode = KQikSoftkeyStyleSmallPortrait;
            command_list = r_Hello_commands;
            view = r_Hello_layout;
            }
        };
    }
```

Kuva 11 *Näkymän tukemien UI-konfiguraatioiden määrittely*

Yllä olevassa kuvassa on tavallisen graafisen UIQ-sovelluksen tukemien käyttöliittymäkonfiguraatioiden esittelyt. Eri konfiguraatiot määritellään QIK_VIEW_CONFIGURATION-määrittelyn avulla. Jokaiselle konfiguraatiolle voidaan tässä määritellä, mitkä komennot ja mikä näkymä on käytössä milläkin konfiguraatiolla. Yllä olevassa esimerkissä käytetään kaikilla konfiguraatioilla samoja komentoja ja samaa näkymää. /11/

Jos puhelin käyttää käyttöliittymäkonfiguraatiota, jolle ei ole sovelluksen resursseissa määrittelyä, etsii järjestelmä sopivimman määritellyn konfiguraation ja käyttää sitä sovelluksen alustamiseen. Sovelluksen on mahdollista pyytää järjestelmää vaihtamaan haluttuun konfiguraatioon mutta tämä ei välttämättä ole mahdollista, jos puhelin ei tue kyseistä konfiguraatiota. Järjestelmä määrää siis aina, mitä konfiguraatiota käytetään. Koska järjestelmä etsii aina sopivimman konfiguraation, ei sovelluksessa tarvitse huolehtia kaikista konfiguraatioista, vaan sama koodi toimii käytännössä kaikissa puhelimissa. /11/

5 WIRELESS TEST FRAMEWORK

Wireless Test Framework on Sasken Finland Oy:ssä Symbian-alustalle diplomi- ja opinnäytetöinä kehitetty testaustyökalu. Sen avulla voidaan tehdä yksikkö-moduuli- ja integrointitestausta. Sillä voidaan siis testata esimerkiksi jonkun luokan toimintaa ja varmistaa, että toiminta on haluttua.

Testikehykseen on aiemmin tehty käyttöliittymä S60-alustalle. Sovelluksen toiminta haluttiin laajentaa toimimaan myös UIQ-alustalla, joka oli tämän tutkintotyön tarkoitus. Koska S60- ja UIQ-alustat pohjautuvat Symbian-käyttöjärjestelmään, oli suurin työ siirtää käyttöliittymä UIQ:lle, johtuen UI:ssa käytettävistä alustojen omista luokista. Seuraavissa kappeleissa tutustutaan hieman tarkemmin testikehykseen ja erityisesti sen käyttöliittymään.

5.1 Yleistä testikehyksestä

Testikehyksen tärkeimmän toiminnallisuuden toteuttaa TestServer-komponentti, jota käytetään asiakas-palvelin-periaatteella. TestServer toteuttaa esimerkiksi testimoduulien ja –tapausten latauksen, testien ajamisen ja tulosten raportoinnin. Testitapaukset sijaitsevat testimoduuleissa, jotka ovat dll-tyyppisiä kirjastoja. Yksi kirjasto voi sisältää esimerkiksi jonkun tietyn luokan testaamiseen kirjoitettuja testitapauksia.

TestServerin toimintaa ohjataan käyttöliittymän avulla. Käyttäjä voi valita haluamansa testitapaukset haluamistaan testikirjastoista ja muodostaa näin testisetin. Kun testejä ajetaan, ilmoittaa TestServer testien etenemisestä tarkkailijaluokalleen. Kun tarkkailijaluokka saa tiedon esimerkiksi testitapauksen ajon valmistumisesta, päivittää se näkymän vastaamaan tätä tilannetta.

Testikehykseen on tehty TestServerin ja UI:n lisäksi muitakin osia, jotka ovat ScriptingModule ja SynchronizationModule. Näiden moduulien avulla testitapauksia voidaan skriptata ja synkronoida, mutta testien ajaminen onnistuu ilmankin näiden moduulien käyttöä. Tässä tutkintotyössä ei käsitellä muita testikehyksen osia kuin käyttöliittymää ja sen kommunikointia TestServerin kanssa.

5.2 Käyttöliittymä

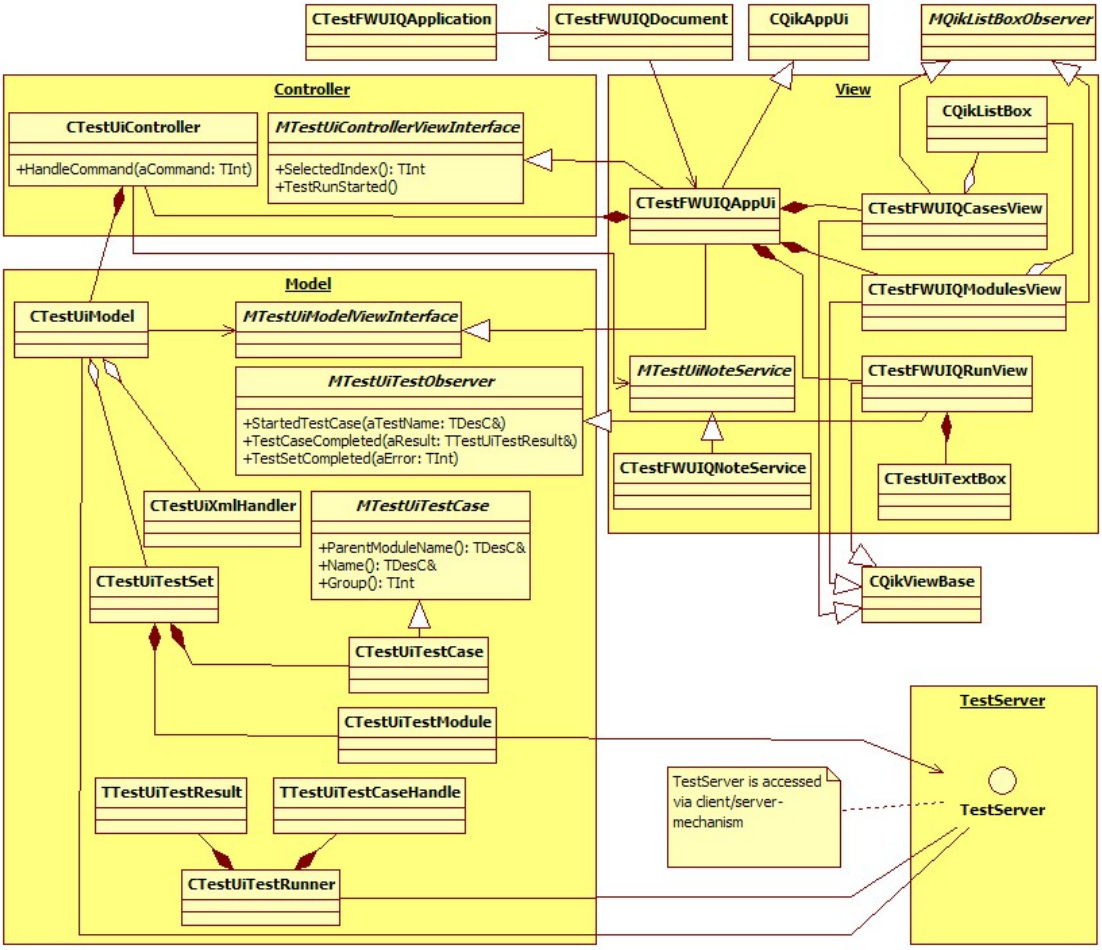
Testikehyksen käyttöliittymän UIQ-versio perustuu sovelluksen S60-versioon. Täten käyttöliittymä on saman tyyppinen ja sisältää samat ominaisuudet molemmilla alustoilla. Seuraavissa kappaleissa perehdytään tarkemmin käyttöliittymän rakenteeseen ja toimintaan.

5.2.1 Rakenne

Käyttöliittymä noudattaa MVC-arkkitehtuuria ja sisältää useita luokkia, jotka muodostavat malli-, näkymä- ja kontrollerimoduulit. Kontrolleri käsittelee kaikki näkymiltä tulevat komennot ja ohjaa mallin toimintaa niiden perusteella. Kontrolleri pitää huolen myös näkymien päivittämisestä tarpeen mukaan.

Malli sisältää tiedon sovelluksen tilasta ja kommunikoi käyttöliittymän ulkopuolelle testitapausten ja -moduulien käsittelyn yhteydessä. Kontrolleri voi mallin avulla esimerkiksi lisätä testimoduuleja ja –tapauksia, ajaa testitapauksia ja poistaa moduuleja ja tapauksia. Näkymätkin ovat yhteydessä malliin vakioviitteen kautta. Näkymät voivat kysyä mallilta esimerkiksi testisetin sisällön, jonka avulla ne päivittävät itsensä vastaamaan sovelluksen tilaa.

Näkymiä käyttöliittymässä on kolme, joista kerrotaan tarkemmin seuraavissa kappaleissa. Kuvassa 12 on nähtävissä käyttöliittymän luokkakaavio.



Kuva 12 *Käyttöliittymän luokkakaavio*

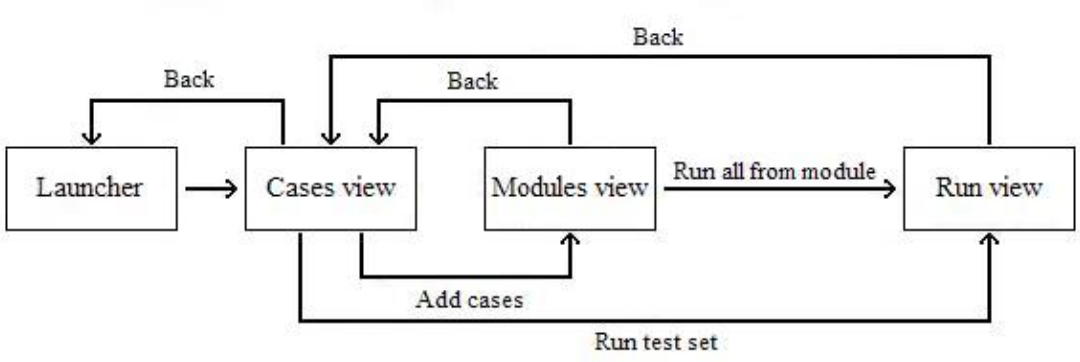
Sovelluksen AppUi-luokka suorittaa lähinnä vain näkymien hallintaa ja välittää käyttäjän komennot kontrollerille käsiteltäväksi. AppUi toteuttaa MTestUiControllerViewInterface- ja MTestUiModelViewInterface-rajapinnat, joiden avulla kontrolleri ja malli ilmoittavat näkymämoduulille tapahtumista. Esimerkiksi Update-funktiossa AppUi välittää päivityskäskyn aktiivisena olevalle näkymälle, jolloin käyttäjälle näkyvä näkymä saadaan päivitettyä.

5.2.2 Näkymät sekä toiminnallisuudet

Testikehyksen käyttöliittymä koostuu siis kolmesta näkymästä. Näkymät ovat tapausnäkymä, moduulinäkymä sekä ajonäkymä. Koska käyttöliittymä suunniteltiin kosketusnäyttöä hyödyntäville laitteille, on näkymien suunnittelussa otettu huomioon käytön helppous sekä kosketusnäytön tuomat uudet mahdollisuudet.

Käyttöliittymän suunnittelussa lähdettiin siitä, että toimintojen valinnat pitäisi pystyä tekemään helposti ja valikoiden käyttöä pitäisi saada rajoitettua. Eniten toimintoja on sovelluksen moduuli- ja tapausnäkymissä. Nämä näkymät ovat saman tyyppiset, sisältäen listat, jotka kuvaavat ladattuja moduuleja ja testisettiä. UIQ:n monipuolinen listaluokka CQikListBox tarjosi mahdollisuuden sisällyttää toimintoja suoraan listan alkioiden yhteyteen, sikäli kun toiminto kohdistui johonkin tiettyyn moduuliin tai testitapaukseen. Tämän avulla sovelluksen käyttöä saatiin helpommaksi, kun kaikkia valintoja ei tarvitse tehtä valikon kautta. Komentojen käytöstä listan kautta kerrotaan lisää myöhemmin.

Koska sovelluksessa on kolme näkymää, piti suunnitella järkevin tapa navigoida näkymien välillä back-näppäintä käyttäen. Koska tapausnäkymän kautta voidaan ladata testisettejä sekä aloittaa setin ajaminen, voidaan sitä pitää sovelluksen päänäkymänä. Kuvassa 13 on esitetty testikehyksen näkymien välillä navigointi.



Kuva 13 *Testikehyksen näkymien välillä navigointi*

Sovelluksen käynnistyessä tapausnäkymä on aktiivinen. Sovelluksen muista näkymistä palataan back-näppäimellä aina kyseiseen näkymään. Kuvassa 13 on nähtävissä myös komentoja, joiden seurauksena näkymiä vaihdetaan. Näistä

komennoista ja näkymien toiminnallisuuksista sekä kommunikoinnista muiden luokkien kanssa kerrotaan lisää seuraavissa kappeleissa.

5.2.2.1 Tapausnäkymä

Tapausnäkymä sisältää listan tämän hetkisen testisetin testitapauksista. Näkymä antaa mahdollisuuden ajaa testisetin, poistaa testitapauksia, ladata tallennetun testisetin tai siirtyä moduulinäkymään. Kun tapausnäkymään siirrytään, kysyy näkymä mallilta nykyiseen testisettiin lisätyt testitapaukset ja päivittää listan vastaamaan niitä. Jokainen listan alkio vastaa yhtä testitapausta. Kuvassa 14 on nähtävissä tapausnäkymä tilanteessa, jossa testisetti sisältää viisi testitapausta.

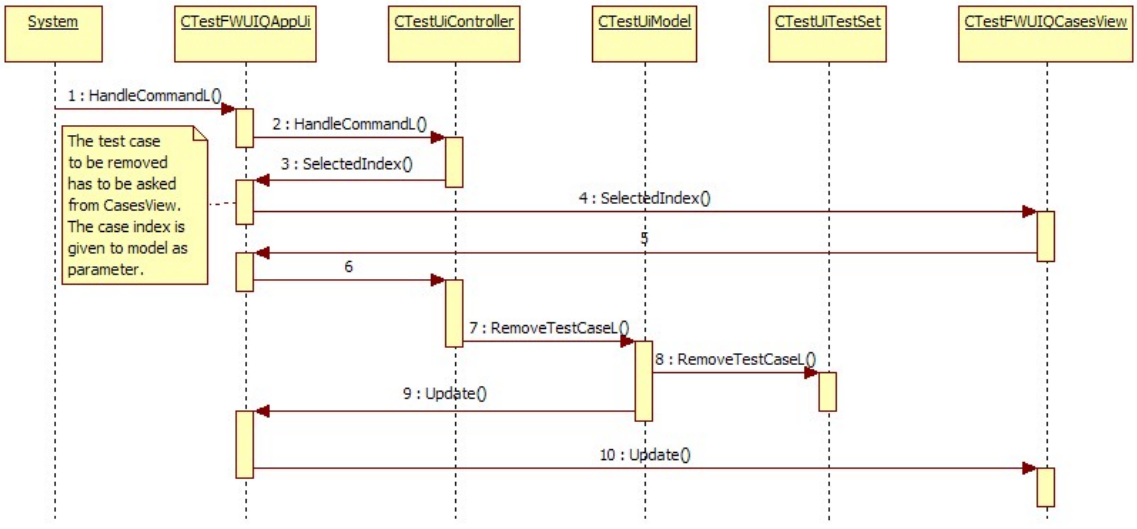
Kuva 14 *Tapausnäkymä, kosketusnäyttöä hyödyntävä toimintonäppäintyyli*

Listan alkiot sisältävät kaksi riviä, joista ylempi kertoo testitapauksen nimen ja toinen toiminnon, joka suoritetaan, kun listan alkio valitaan. Kun listan alkio valitaan, poistetaan kyseinen testitapaus testisetistä. Kuten kuvasta 14 näkyy, on listan alkioita mahdollista valita useita kerralla mutta toteutusta usean tapauksen valikoivaan poistoon ei ole vielä tehty. Vaikka testitapauksia ei voi poistaa valikoivasti, on valikossa kuitenkin mahdollisuus poistaa kaikki testitapaukset

kerralla. Listan alkion kosketus käsitellään näkymän HandleListBoxEventL-funktiossa, koska näkymä on rekisteröity kyseisen listan tarkkailijaksi.

HandleListBoxEventL-funktiossa luodaan ja laukaistaan testitapauksen poistoa vastaava näppäintapahtuma eli periaatteessa toimitaan samoin kuin FEP:t, jotka tekevät esimerkiksi kosketustapahtumalle esikäsittelyä ja luovat sitten sopivan näppäintapahtuman. Näppäintapahtuma käsitellään AppUi:n HandleCommandL-funktiossa.

UIQ:ssa näppäintapahtumaa kuvastaa CQikCommand-tyyppinen olio, joka poikkeaa S60-alustan käyttämästä TInt-tyypistä. UIQ:n komentotyyppi sisältää enemmän tietoja komennosta, kuten tyyppitiedon, joka määrittää, missä komento näytetään milläkin käyttöliittymäkonfiguraatiolla. Koska eri alustat käyttävät erityyppisiä komentoja, on UIQ-version AppUi:n muunnettava komento TInt-tyyppiseksi, jolloin kontrolleriin ei vaadita muutoksia komentojen käsittelyn suhteen, vaan sama toteutus toimii eri alustoilla. Kuvassa 15 on nähtävissä yksinkertaistettu testitapauksen poistosekvenssi.



Kuva 15 *Testitapauksen poistaminen*

Testitapauksen poisto etenee seuraavasti: AppUi saa näkymässä luodun näppäinkomennon käsiteltäväkseen ja välittää sen kontrollerille. Kontrolleri kysyy AppUi:lta valitun testitapauksen indeksin ja käskee tämän jälkeen mallia

poistamaan testitapauksen. Kuten kappaleessa 5.2.1 kerrottiin, AppUi toteuttaa sekä MTestUiModelViewInterface- että MTestUiControllerViewInterface-rajapinnat ja toimii täten rajapintana näkymien ja kontrollerin sekä mallin välillä. AppUi:lla on tiedossa aktiivisena oleva näkymä, jolle se välittää SelectedIndex-kutsun.

Malli välittää testitapauksen poistopyynnön CTestUiTestSet-luokalle, joka poistaa tapauksen. Tämän jälkeen näkymä pitää päivittää, koska testitapaus on poistettu. Näkymien päivitys tapahtuu samaan tyyliin, kuin indeksin kysely eli AppUi kutsuu aktiivisen näkymän Update-funktiota, jossa näkymä päivitetään kysymällä tarvittavat tiedot mallilta.

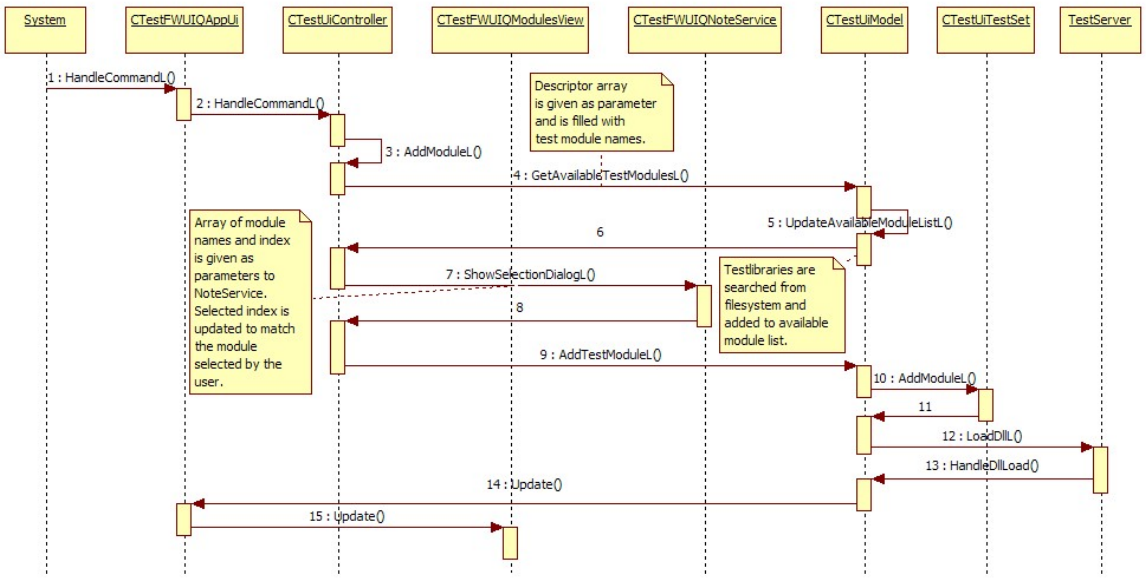
5.2.2.2 Moduulinäkymä

Moduulinäkymässä on lista testikehykseen ladatuista testimoduuleista. Näkymässä on mahdollista ladata lisää testimoduuleja, poistaa moduuleja, lisätä testisettiin testitapauksia, ajaa kaikki tapaukset halutusta moduulista ja siirtyä tapausnäkymään.

Kuten tapausnäkymässäkin, on toimintoja sisällytetty listan alkioiden yhteyteen, jotka vastaavat ladattuja moduuleja. Moduulin poisto, testitapauksen lisäys ja kaikkien tapausten ajo tietystä moduulista on mahdollista tehdä suoraan listan avulla, ilman valikosta tehtäviä valintoja. Moduulista on myös mahdollista lisätä kaikki testitapaukset testisettiin kerralla. Kuten tapausnäkymässäkin, myös moduulinäkymässä listan alkioiden toinen rivi määrää, mikä toiminto tehdään, kun alkiota kosketetaan. Haluttua toimintoa voidaan muuttaa alkion toisen rivin päädyissä olevia nuolia koskettamalla. Kuvassa 16 on nähtävissä moduulinäkymä, kun testikehykseen on ladattu kaksi testimoduulia.

Kuva 16 *Moduulinäkymä, kosketusnäyttöä hyödyntävä toimintonäppäintyyli*

Uuden testimoduulin lataaminen tapahtuu valitsemalla valikosta testimoduulin lataamistoiminto. Valikoista tehtäville komennoille ei tehdä näkymissä mitään käsittelyä vaan annetaan AppUi:n käsitellä ne ja välittää komento kontrollerille. Kuvassa 17 on nähtävissä testimoduulin lataamissekvenssi.



Kuva 17 *Testimoduulin lataaminen*

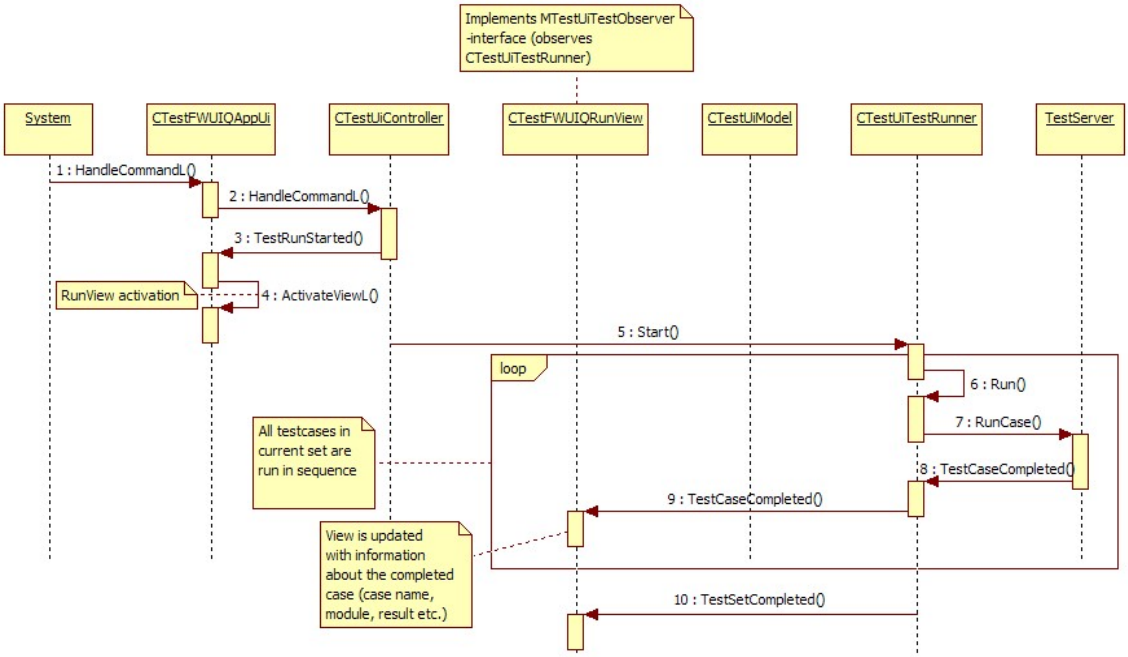
Testimoduulin lataaminen etenee seuraavasti: AppUi välittää komennon kontrollerille, joka kysyy saatavilla olevia testimoduuleja mallilta. Kontrolleri antaa mallille parametrina muuttujan, johon malli lisää saatavilla olevien testimoduulien nimet. Malli päivittää moduulilistan etsimällä dll-tyyppisiä testikirjastoja tietystä paikasta.

Kun kontrolleri on saanut tietoonsa saatavilla olevat moduulit, kutsuu se NoteServicen ShowSelectionDialogL-funktiota. NoteService näyttää käyttäjälle dialogin, joka sisältää listan moduuleista. Kun käyttäjä valitsee haluamansa moduulin, asetetaan kyseisen moduulin indeksi parametrina saatuun muuttujaan.

Tämän jälkeen kontrolleri käskee mallia lisäämään kyseisen moduulin antamalla valitun indeksin parametrina. Malli kutsuu TestSet-luokan AddModule-funktiota, jossa uusi moduuli lisätään testisettiin. Seuraavaksi pitää tehdä moduulin varsinainen lataus, joka tehdään TestServerin avulla. TestServer on aktiiviolio, joka latauksen valmistuttua kutsuu mallin HandleDllLoad-funktiota. Jos moduulin lataus onnistui, päivitetään näkymä. Jos lataus ei onnistunut, poistetaan moduuli testisetistä.

5.2.2.3 Ajonäkymä

Ajonäkymä sisältää testikehykseen itse tehdyn tekstikentän, johon ajettavien testien tulokset tulostetaan. Näkymässä ei tällä hetkellä ole muita toimintoja, kuin testisetin ajamisen uudelleen käynnistys, sekä siirtyminen takaisin tapausnäkymään. Ajonäkymään siirtyminen, sekä testien ajaminen on kuvattu kuvassa 18.



Kuva 18 *Testitapausten ajaminen*

Testisetin ajaminen etenee seuraavasti: Kontrolleri saa käsiteltäväkseen testisetin ajamiskomennon, jonka käyttäjä on laukaissut tapausnäkymästä tai ajonäkymästä. Sama sekvenssi toistuu myös silloin, kun käyttäjä valitsee ”Run all from module”-komennon moduulinäkymässä. AppUi saa tiedon testien ajamisen aloittamisesta kontrollerilta ja aktivoi ajonäkymän, johon testien ajon aikana tulostetaan tietoja.

Seuraavaksi kontrolleri kutsuu CTestUiTestRunner-tyyppisen olion Start-funktiota, jossa olio asetetaan aktiiviseksi ja käsketään TestServeriä ajamaan testitapaus. Kun testitapaus on ajettu, saa CTestUiTestRunner siitä tiedon ja kutsuu tarkkailijaluokkansa TestCaseCompleted-funktiota. Kyseisen olion tarkkailijana toimii ajonäkymän ilmentymä, joka päivittää itsensä aina tapauksen ajon valmistuttua. Käytännössä näkymä tulostaa tekstikenttäänsä ajetun testitapauksen nimen ja tiedon siitä, onnistuiko testi vai ei. Kaikkien testisetissä olevien testitapausten ajo tehdään edellä kuvatun mukaisesti peräkkäin.

Kun testisetin kaikki testitapaukset on ajettu, saa ajonäkymä tästä tiedon ja tulostaa näytölle testien ajamisen päättyneen. Lisäksi näytölle tulostetaan läpi menneiden ja

epäonnistuneiden testitapausten määrät. Kuvassa 19 on nähtävissä ajonäkymä, kun kaikki viisi testisetissä ollutta testitapausta on ajettu.

Kuva 19 *Ajonäkymä, kosketusnäyttöä hyödyntävä toimintonäppäintyyli*

5.2.3 Käyttöliittymän parannukset

Sen lisäksi, että tässä opinnäytetyössä tuli hyödyntää UIQ-alustan ja kosketusnäytön mukanaan tuomia mahdollisuuksia käyttöliittymän suunnittelussa, piti sovelluksen käyttöliittymää parantaa myös joidenkin käyttöä helpottavien lisäysten avulla. Käytännössä suurin osa lisäyksistä on hyvin pieniä mutta ne parantavat käyttäjäystävällisyyttä huomattavasti. Näitä pieniä parannuksia ovat:

- moduulista on nyt mahdollista lisätä kaikki testitapaukset testisettiin kerralla
- testisetistä voidaan poistaa kaikki testitapaukset kerralla
- testisetin ajaminen voidaan käynnistää uudestaan suoraan ajonäkymästä

Edellä mainittujen parannusten lisäksi toteutettiin käyttöliittymään testisettien lataus xml-muotoisesta tiedostosta. Kun testisettien lataus tehdään yksinkertaiseen xml-tiedostoon perustuen, on testaajan helppo luoda haluamiaan testisettejä

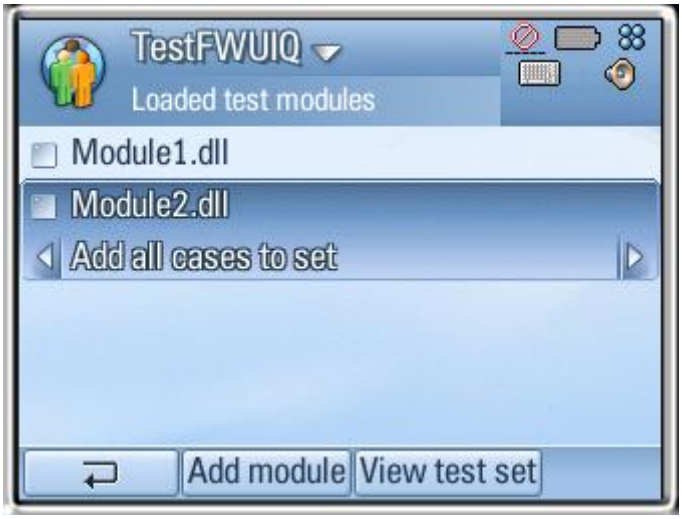
tietokoneella ja käyttää niitä sitten puhelimessa. Xml-tiedostoon voidaan tallentaa käytännössä rajaton määrä testisettejä, jotka voivat sisältää testitapauksia useista eri moduuleista. Testisetistä kuvataan xml-tiedostossa setin nimi, testisettiin kuuluvien testitapausten nimet sekä moduulit, joihin testitapaukset kuuluvat.

5.2.4 Toiminta eri UI-konfiguraatioilla

Eri käyttöliittymäkonfiguraatioita varten ei tässä sovelluksessa ole tehty mitään muokkauksia, vaan kaikilla konfiguraatioilla käytetään samoja komentoja ja näkymiä. Optimointiin eri konfiguraatioita varten ei nähty tarvetta, koska sovelluksen käyttöliittymä toimi kaikilla testatuilla konfiguraatioilla hyvin emulaattorilla testattaessa.

Moduuli- ja testitapausnäkymät koostuvat CQikListBox-tyyppisestä listasta, joka on hyvin muuntautuva ja suunniteltu toimimaan eri käyttöliittymäkonfiguraatioilla. Testitapausten ajonäkymässä on käytössä testikehykseen itse tehty tekstikontrolli. Myös tämä kontrolli mukautuu hyvin eri konfiguraatioihin. Alla olevissa kuvissa 20, 21 ja 22 on kuvakaappauksia sovelluksen ajamisesta kynätyylin ollessa käytössä. Kuvien tilanteissa näyttöä käytetään vaakasuunnassa. Kuvat vastaavat kappaleissa 5.2.2.1, 5.2.2.2 ja 5.2.2.3 esitettyjä tilanteita.

Kuva 20 *Tapausnäkymä, kynätyyli, vaakasuuntainen näyttö*



Kuva 21 *Moduulinäkymä, kynätyyli, vaakasuuntainen näyttö*

Kuva 22 *Ajonäkymä, kynätyyli, vaakasuuntainen näyttö*

Kuten kuvista nähdään, kynätyyliä käytettäessä valikon komennot on sijoitettu Button Bariin. Tämä on toimiva ratkaisu, koska toiminnot saa tehtyä helposti ilman valikoita.

5.3 Ongelmakohdat

Tässä kappaleessa perehdytään joihinkin työn tekemisessä esiintyneisiin ongelmiin ja esitetään ratkaisut niiden selvittämiseksi.

5.3.1 Käytöstä poistettu dialogi

Sovelluksen S60-versiossa käytetään CAknSelectionListDialog-tyyppistä oliota valintalistan toteutukseen. Kyseinen luokka periytyy CAknDialog-luokan kautta CEikDialog-luokasta, jota ei UIQ:n 3.0-versiossa enää tueta.

CEikDialogin korvaajiksi UIQ:n 3.0-versiossa tuli CQikSimpleDialog ja CQikViewDialog. CQikViewDialog on tavallisen näkymän ja dialogin yhdistelmä. Se mahdollistaa dialogien käytöstä tutun synkronisen koodin suorituksen sekä näkymien tuen monipuolisille konfiguraatioille. Synkroninen koodin suoritus tarkoittaa sitä, että koodin suoritus jää odottamaan dialogista paluuta. /4/

Sovelluksen valintalista toteutettiin periyttämällä luokka CQikViewDialogista sekä MQikListBoxObserverista. Dialogi sisältää CQikListBox-tyyppisen listan, johon halutut listan alkiot lisätään. Luokan rakentajalle annetaan parametrina taulukko listan alkioista, joilla luotava lista täytetään. Kun käyttäjä valitsee listalta jonkun alkion, palauttaa dialogi valitun alkion indeksin kutsujalle.

5.3.2 UIQ yhtiön konkurssi

UIQ yhtiö hakeutui konkurssiin tämän työn tekemisen aikana. Konkurssin seurauksena esimerkiksi developer.uiq.com-sivusto lopetti toimintansa. Sivusto oli saman tapainen kuin Forum Nokian sivut eli se sisälsi dokumentaatiota sekä keskustelualueita sovelluskehitykseen liittyen.

Koska UIQ-alustan käyttäjä- ja kehittäjäkunta on selvästi pienempi kuin S60-alustalla, oli konkurssin jälkeen hankala löytää apua Internetistä. Erityisesti sovelluskehittäjien keskustelupalstoja oli vaikea löytää. UIQ:n SDK:n dokumentaatio on kuitenkin hyvää ja kattavaa.

UIQ:n konkurssi ei tarkoita sitä, että kaikki UIQ-alustan hyvät ominaisuudet menisivät hukkaan. Nokia, Sony Ericsson, Motorola, NTT DOCOMO ja muutama muu yritys ovat perustaneet Symbian Foundation-säätiön, jossa yhdistyvät S60-,

UIQ- ja MOAP-alustat. Yritykset ovat siis lupautuneet antamaan omat ohjelmistoalustansa säätiön käyttöön. Nokia antaa säätiön käyttöön myös omistamansa Symbian-käyttöjärjestelmän. Yritykset kehittävät yhdessä tätä uutta alustaa, johon liitetään ominaisuuksia kaikista mainituista alustoista. Uusi Symbian-alusta on tarkoitus saattaa avoimeksi lähdekoodiksi kahden vuoden kuluessa, jolloin lähdekoodit ovat kaikkien saatavilla. Voidaan siis olettaa, että dokumentaatiosta ja kehittäjäyhteisöistä ei jatkossa tule olemaan puutetta. /13/

5.4 Työn onnistuminen

Työn tavoitteena oli saada testikehys toiminaan UIQ-alustalla kosketusnäyttöä hyödyntäen. Käyttöliittymä piti suunnitella mahdollisimman helppokäyttöiseksi ja esimerkiksi valikoiden liiallista käyttöä tuli välttää. Lisäksi käyttöliittymään tuli tehdä joitakin parannuksia ja lisäominaisuuksia.

Ottaen huomioon sen, että minulla ei ollut yhtään kokemusta UIQ-alustasta, eikä sen enempää kosketusnäytön hyödyntämisestä Symbianiin pohjautuvissa laitteissa ennen tämän työn tekemistä, onnistui työ mielestäni hyvin. Käyttöliittymä UIQ-sovelluksessa on mielestäni helppokäyttöinen ja toimiva.

Lisäominaisuuksista testisettien lataus xml-tiedoston perusteella on mielestäni hyvä lisäys, joka helpottaa ja nopeuttaa sovelluksen käyttöä huomattavasti. Koska ennen ei ollut mahdollisuutta ladata testisettejä, jouduttiin moduulit ensin lataamaan sovellukseen ja sieltä lisäämään halutut testitapaukset testisettiin. Nyt testisetti voidaan määritellä xml-tiedostoon ja sitten vain ladata se. Testisetin latauksen yhteydessä moduulit ja testitapaukset ladataan automaattisesti, käyttäjän tarvitsee vain valita ladattava setti.

5.5 Jatkokehitysajatuksia

Eräs jatkokehitysajatus on testisetin tallennus xml-tiedostoon. Tämän toteuttaminen ei olisi kovin vaativaa, koska xml-tiedoston muoto on jo määritelty. Xml-tiedosto on hyvin yksinkertainen ja tiedot, jotka tiedostoon pitää tallentaa, on helposti

saatavilla. Käytännössä tämän toiminnon lisääminen vaatisi lisäyksiä kontrolleriin, malliin sekä CTestUiXMLHandler-luokkaan, jossa tiedot pitäisi kirjoittaa tiedostoon.

Toisena jatkokehitysajatuksena on kehittää edelleen käyttöliittymän käytännöllisyyttä. Sekä tapaus- että moduulinäkymän listoissa on UIQ-versiossa mahdollisuus valita useita alkioita kerralla. Useiden alkioiden valinta ei tällä hetkellä kuitenkaan vaikuta mihinkään, koska toteutus on tekemättä. Tapausnäkymässä useiden testitapausten valikoiva poisto olisi selkeä parannus ja luultavasti mahdollista toteuttaa melko pienellä työllä. Tällä hetkellähän testitapauksia voi poistaa yksitellen tai kaikki kerralla. Moduulinäkymässä useiden alkioiden valinta olisi myös hyödyllistä joidenkin toimintojen kohdalla. Esimerkiksi usean moduulin poisto kerralla tai kaikkien testitapausten lisäys testisettiin useasta moduulista kerralla parantaisi käyttöliittymän käyttäjäystävällisyyttä entisestään.

LÄHDELUETTELO

Painetut lähteet

5 Harrison, Richard; Shackman, Mark, *Symbian OS C++ for Mobile Phones*, Volume 3. Wiley, 2007.

Sähköiset lähteet

1 UIQ Technology AB, *The basics in Application development tutorial*. UIQ Technology AB, 2007. [www-sivu]. [viitattu 7.10.2008]. Saatavissa: http://developer.uiq.com

2 Nokia, *S60 5th Edition C++ Developer's Library v1.0*. Nokia, 2008. [www-sivu]. [viitattu 13.10.2008]. Saatavissa: http://www.forum.nokia.com

3 UIQ Technology AB, *The UIQ 3 Application Behaviour Guide*. UIQ Technology AB, 2007. [pdf-dokumentti]. [viitattu 20.10.2008]. Saatavissa: http://developer.uiq.com

4 UIQ Technology AB, *Views and dialogs – UIQ Books*. UIQ Technology AB, 2008. [www-sivu]. [viitattu 9.2.2009]. Saatavissa: http://books.uiq.com/index.php

6 Nokia, *S60 5th Edition: What's New For Developers*. Nokia, 2008. [pdf-dokumentti]. [viitattu 11.2.2009]. Saatavissa: http://www.forum.nokia.com

7 UIQ Technology AB, *UIQ releases*. UIQ Technology AB, 2008. [www-sivu]. [viitattu 12.2.2009]. Saatavissa: http://www.uiq.com

8 Symbian Software Limited, *Essential UIQ Getting Started*. Symbian Software Limited, 2006. [pdf-dokumentti]. [viitattu 12.2.2009]. Saatavissa: https://developer.symbian.com

9 Symbian Software Limited, *Front End Processor Overview*. Symbian Software Limited, 2007. [www-sivu]. [viitattu 19.2.2009]. Saatavissa: http://www.symbian.com/

10 UIQ Technology AB, *Understanding UI components – UIQ Books*. UIQ Technology AB, 2008. [www-sivu]. [viitattu 24.2.2009]. Saatavissa: http://books.uiq.com/index.php

11 UIQ Technology AB, *UIQ 3 – "View and Command"*. UIQ Technology AB, 2007. [pdf-dokumentti]. [viitattu 29.2.2008]. Saatavissa: http://developer.uiq.com

12 UIQ Technology AB, *Views and dialogs – UIQ Books*. UIQ Technology AB, 2008. [www-sivu]. [viitattu 25.2.2009]. Saatavissa: http://books.uiq.com/index.php

13 Nokia, *Matkaviestintäalan johtavat yritykset yhtenäistävät ja avaavat Symbianin ohjelmistoalustan*. Nokia, 2008. [www-sivu]. [viitattu 20.4.2009]. Saatavissa: http://www.nokia.fi