MIKKELIN AMMATTIKORKEAKOULU

MIKKELIN TEKNILLINEN OPPILAITOS

Sähkötekniikan koulutusohjelma

Insinöörityö

FITWARE-ANALYYSIOHJELMAN SUUNNITTELU JA OHJELMOINTI FIT-TEST OY:LLE

Mika Ahvenranta

1997

Työn valvoja: Markku Nuutinen

Työn ohjaaja: Hannu Rahikainen

Hyväksytty arvosanalla _____(___)

___.___.199_____

TIIVISTELMÄ

Työni tarkoituksena oli toteuttaa FitWare-analyysiohjelma Windows-ympäristöön Mikkeliläiselle Fit-Test Oy:lle. FitWare-ohjelma tarkentaa ja tekee helpommaksi ihmisen kestävyyskunnon mittauksen, ja sen avulla voidaan seurata myös työpäivän ja harjoituksen fyysistä kuormittavuutta. Ohjelma käyttää analyyseissään hyväksi polkupyöräergometriä ja Polar Electro Oy:n tietokonepurettavia sykemittareita.

Työn tehtäviin kuuluivat tiedon hankkiminen vaatimusmäärittelyä varten, ohjelman suunnittelu ja ohjelman koodaus. Ohjelmistokehityksen muut vaiheet, kuten testaus ja ylläpito eivät kuuluneet tehtäviin.

Aloitin työn keräämällä käyttäjältä ja käytettäviltä asiantuntijoilta tietoa ohjelmiston vaatimusmäärittelyä varten. Vaatimusmäärittelyn valmistuttua, ohjelma mallinnettiin Microsoftin Visual Basic-kehitintyökalun avulla. Mallinnuksella paikattiin vaatimusmäärittelyn puutteita ja hahmoteltiin lopullisen ohjelman ulkonäkö ja toiminnot. Mallinnusta käytettiin useaan otteeseen, jotta ohjelman ulkonäkö ja toiminnot saatiin hiottua halutuiksi. Valitsin Visual Basicin myös FitWare-ohjelman ohjelmointikieleksi, jonka avulla muutosten tekeminen ohjelmaan oli helppoa ja nopeaa.

Työn tuloksena syntyi valmis tuote Fit-Test Oy:lle. FitWare-ohjelmaa käytetään tällä hetkellä mm. työterveysasemilla, kuntoutuslaitoksissa ja puolustusvoimissa. Työstä kertyi työnantajalle myös arvokasta tietoa ohjelmakehityksen eri vaiheiden ajankäyttöön ja kustannuksiin liittyvistä asioista FitWare-ohjelman jatkokehittelyä varten.

ABSTRACT

The purpose of this design was to create a Windows based analysis and follow-up programme called FitWare. It can be used for analysing physical exercise capacity by using bicycle ergometer and heart rate monitors and interface devices by Polar Electro Ltd. Effects of exercise and physical occupational stress can also be analysed.

I began the work by picking up the information for the programme by interviewing the employer and specialists available. After the requirement specification was completed, I started software modelling with Microsoft's Visual Basic development tool. Modelling the software was essential at this point, because designing the graphical user interface and functions of FitWare programme could be made by modelling. The missing features of the requirement specification were also added by modelling the software. Changes were made all the time during the development of FitWare programme, therefore I chose Visual Basic also as final programming tool for FitWare.

As a result of my work, a complete product, FitWare analysis programme, was created for Fit-Test Ltd. Today FitWare programme is used, for example, in occupational health care centers, rehabilitation centers and also in Finnish armed forces. The emloyer also got valuable information of matters concerning timing and financing of software development by this work. This information facilitates the future planning of FitWare programme.

SISÄLTÖ

1 JOHDANTO 6

2 OHJELMISTOTEKNIIKKA 7

2.1 Ohjelmiston kehityksen perusongelmat 7

2.2 Ohjelmiston elinkaari ja kehitysmallit 9

2.2.1 Käyttäjän vaatimusten määrittely 11

2.2.2 Ohjelmistovaatimusten määrittely 12

2.2.3 Rakenteellinen suunnittelu 15

2.2.4 Yksikkösuunnittelu, koodaus ja läpikäynnit 17

2.2.5 Testaus 18

2.2.6 Katselmukset 19

2.3 Katsaus olio-ohjelmointiin 19

2.3.1 Yleisimmät Windows-ohjelmointivälineet 22

2.3.2 Olio-ohjelman suunnittelumenetelmät 24

3 FITWARE-JÄRJESTELMÄN MÄÄRITTELY 24

3.1 FitWare-järjestelmän teoreettinen tausta 24

3.2 Toiminnallinen kuvaus 26

3.3 Järjestelmän osat 27

3.4 Järjestelmän kohderyhmät 30

4 FITWARE-OHJELMAN VAATIMUSMÄÄRITTELY 30

4.1 Käyttäjät 32

4.2 Toiminnalliset vaatimukset 32

4.2.1 Ergometritestianalyysin toimintojen ominaisuudet 34

4.2.1.1 Sykkeiden syöttäminen 35

4.2.1.2 Tulosten laskeminen 35

4.2.1.3 Sykekäyrän piirtäminen 37

4.2.1.4 Tulokset käyttäjän ohjauksella 38

4.2.1.5 Kestävyysalueet 39

4.2.1.6 Liikuntaohjeet 39

4.2.1.7 Tulosten vertaaminen normiin 39

4.2.1.8 Oheismittaukset 40

4.2.1.9 Tulosten vertaaminen edellisiin testeihin 42

4.2.2 Työpäivän ja harjoituksen fyysisen kuormittavuuden analyysien toimintojen ominaisuudet 42

4.2.2.1 Sykkeiden syöttäminen 42
4.2.2.2 Tulosten laskeminen 43
4.2.2.3 Sykekäyrän piirtäminen 44
4.2.2.4 Tulokset käyttäjän ohjauksella 45
4.2.2.5 Kestävyysalueet 45
4.2.2.6 Energian kulutus ravintoaineina 45
4.2.2.7 Oheismittaukset 45
4.2.3 Muut tarvittavat toiminnot 46
4.3 Liittymät 47
5 FITWARE-OHJELMAN ARKKITEHTUURIN SUUNNITTELU 48
5.1 Sykemittareiden sovitus ohjelmaan 48
5.2 Ohjelmointikielen valinta 49
5.3 Ohjelman mallintaminen 50
5.3.1 Käyttöliittymätyypin valinta 51
5.3.2 Käyttöliittymän ikkunointi 53
5.3.3 Näkyvien olioiden eli komponenttien sijoittelu 56
5.4 Toimintojen ryhmittely 60
6 FITWARE-OHJELMAN YKSIKKÖSUUNNITTELU JA KOODAUS 62
6.1 Ohjelmointi Visual Basicillä 62
6.2 FitWare-ohjelman ohjelmointi 64
7 FITWARE-OHJELMAN KÄYTTÖÖNOTTO JA TULEVAISUUDEN NÄKYMÄT 67
8 MITÄ OPIN TYÖSTÄ? 68

LÄHTEET 71

LIITTEET
1 LIIKUNTALÄÄKE-IKKUNAN OLETUSTEKSTIT FITWARE-OHJELMASSA 72
2 BORG-INDEKSIÄ VASTAAVAT SANALLISET MÄÄRITTELYT 73
3 ENERGIA-IKKUNAN ELINTARVIKKEET JA NIIDEN ENERGIASISÄLLÖT FITWARE-OHJELMASSA 74
4 FITWARE-OHJELMASSA TARVITTAVAT HENKILÖTIEDOT 75
5 FITWARE-OHJELMAN TÄRKEIMMÄT TULOSTEET 77
6 ESIMERKKI FITWARE-OHJELMAN OHJELMAKOODISTA 85

1 JOHDANTO

Polkupyöräergometrejä on käytetty jo 1950-luvulta lähtien ihmisen maksimaalisen hapenkulutuksen mittaamiseen. Polkupyöräergometri muistuttaa ulkonäöltään tavallista kuntopyörää, mutta eroavuutena on se, että poljettavaa vastusta eli kuormittavuutta voidaan säätää huomattavasti tarkemmin. Maksimaalista hapenottokykyä voidaan mitata suoraan uloshengitysilmasta ergospirometrialaitteistoilla tai sitä voidaan arvioida epäsuorasti tavallisemmin sydämen sykintätason noususta kuormitusta lisättäessä. Suorasta testistä saadaan luotettava testitulos, mutta tarvittavan laitteiston kustannukset ovat korkeat. Suoria hapenottotestejä tehdään yleensä vain tutkimuslaitoksissa ja sairaaloissa.

Epäsuorassa ergometritestissä kustannukset ovat pienemmät, mutta testaaja itse joutuu määrittämään rasitus- ja syketiedoista testattavalle maksimaalisen hapenottokyvyn. Ongelmina epäsuorassa ergometritestauksessa on tähän asti ollut kolmiportaisen testimallin käyttö, jossa käytetään vain kolmea kuorma-sykintäpistettä, sekä sykkeiden epätarkka havaitseminen. Kun tämän lisäksi varsinaisen maksimaalisen hapenottokyvyn määrittämiseksi tarvitaan monimutkaisia laskentakaavoja tai jopa silmämääräistä arviointia, mittausvirheiden esiintyminen on hyvin todennäköistä, ja testitulokset eivät ole täysin vertailukelpoisia.

Mikkeliläinen Fit-Test Oy on lähtenyt hakemaan ratkaisua edellä mainittuihin ongelmiin tietotekniikan avulla. Yrityksen tarkoituksena on luoda tietokoneohjelmisto, joka automaattisesti laskee ja analysoi testattavan maksimaalisen hapenottokyvyn epäsuorasta ergometritestistä. Sykkeiden rekisteröintiin käytetään Polar Electro Oy:n tietokonepurettavia sykemittareita, jotka mahdollistavat sykkeiden siirron suoraan tietokoneohjelmaan sarjaliityntäkanavan kautta. Lisäksi ohjelma analysoi myös työpäivän ja harjoituksen fyysistä kuormittavuutta ja pyrkii mittaustuloksien lisäksi tarjoamaan liikuntafysiologista ja -lääketieteellistä perustietoa sekä testattavalle että testaajalle. Järjestelmä on ainutlaatuinen ja se toteutetaan osin yhteistyössä suomalaisten liikuntalääketieteen ja liikuntafysiologian asiantuntijoiden kanssa. Ohjelmiston ensisijaisia käyttäjiä tulee olemaan työterveyshuollon laitokset, kuntoutuslaitokset, testiasemat, sairaalat, terveyskeskukset jne. Tästä eteenpäin tulen käyttämään edellä mainitusta järjestelmästä sen tulevaa ohjelmanimeä FitWare.

Insinöörityöni tavoitteena on tehdä Fit-Test Oy:lle FitWare-ohjelma. Tehtävänäni on toimia ns. tietämysinsinöörinä, joka vastaa järjestelmän rakentamisessa tietämyksen hankkimisesta,

jäsentelystä ja tietokoneelle siirtämisestä. Työnantaja toimii työssä ohjelman käyttäjänä, jota haastattelemalla minun on kerättävä ja jäsenneltävä tarpeeksi tietoa ohjelman vaatimusmäärittelyä varten. Tämän jälkeen tehtävänäni on selvittää, miten saadaan aikaiseksi toimiva tietokoneohjelma, joka täyttää vaatimusmäärittelyn ehdot. Lopuksi kokoan ohjelman näiden tietojen perusteella. Ohjelmistokehityksen muut vaiheet, kuten testaus ja ylläpito eivät kuulu tehtäviini.

Ohjelman toteutuksessa pyrin toteuttamaan hyvän ohjelmiston tunnuspiirteet. Ojanperä määrittelee hyvän ohjelman tuntomerkit seuraavasti:
-ohjelma toimii vaatimusmäärittelyn mukaisesti
-ohjelma on ylläpidettävissä
-ohjelma on enemmän kuin ohjelmakoodi: dokumenttikokonaisuus, joka mahdollistaa kaksi edellistä kohtaa (1, s 146).
Näiden vaatimusten toteuttaminen tulee olemaan ensisijaisen tärkeää, sillä FitWare-ohjelmasta tulee kaupallinen tuote ja sen on kehityttävä asiakkaiden tarpeiden mukaan. Tämän vuoksi ohjelmasta tulee useita eri muunnelmia sekä jatkoversioita, joiden toteuttaminen ei onnistuisi ilman edellä mainittuja vaatimuksia. FitWare-ohjelmiston kaupallisuuden vuoksi insinöörityöstäni puuttuu työnantajan salaisiksi luokitellut tiedot, kuten tarkat moduulikuvaukset ja ohjelmakoodit, joista esitän vain esimerkit.

2 OHJELMISTOTEKNIIKKA

Ohjelmistotekniikkaa tarvitaan ohjelmistohankkeen elinkaaren kaikkien toimintojen läpiviemiseksi ennalta suunniteltujen aikataulujen ja resurssien mukaisesti. Ohjelmistotekniikka on tekniikan lajina vielä varsin nuori. Tämän takia alalla on vieläkin paljon vääriä työskentelytapoja ja myyttejä, jotka aiheuttavat ongelmia ohjelmistojen kehityksessä ja ennen kaikkea laadussa. Ohjelmistojen kohdealueet muuttuvat ja laajenevat hyvin nopeasti, mikä tuo lisäväriä ohjelmistokehityksen alalle - uusia menetelmiä ja kehitystyökaluja otetaan käyttöön.

2.1 Ohjelmiston kehityksen perusongelmat

Laitteiston kehitysprojekti ja ohjelmiston kehitysprojekti ovat molemmat tyypillisiä tekniikan hankkeita, joissa esiintyy samantapainen pääjako: ensin tutkitaan ja määritellään, sitten suunnitellaan ja toteutetaan, ja lopuksi ylläpidetään. Ohjelma on kuitenkin looginen, ei

fyysinen elementti, ja tämän vuoksi se eroaa laitteistosta siten, että
-varsinaista tuotantovaihetta ei ole, vaan kaikki kustannukset liittyvät suunnitteluun ja ensimmäisen kappaleen tekemiseen
-se ei kulu eikä tarvitse varaosia
-ylläpito on vikojen poistamisen lisäksi muutosten ja laajennusten tekoa.

Laitteiston kehityksessä käytettävät komponentit ovat selkeitä rakennusosia, joiden liitännät on standardoitu ja joiden suorituskyky, hinta ja saatavuus tunnetaan tarkasti. Kaikki nämä ominaisuudet kuitenkin puuttuvat ohjelmiston kehityksestä. Tämän vuoksi ohjelmistotekniikan alaa varjostavat monet käytännön ongelmat. Suurimpana ongelmana on, että ohjelmat saadaan aina lopulta toimimaan, mutta aikataulut ja kustannukset poikkeavat merkittävästi suunnitelluista. Ojalehdon mukaan (1, s. 27) yksi syy tähän on se, että vanhoista projekteista ei kerätä systemaattisesti tietoa siitä, kuinka kauan kukin työ kestää - aikataulut ovat arvausta. Muita syitä Ojalehdon mukaan ovat puutteelliset yhteydenpidot tekijän ja tarvitsijan välillä, testauksen ja katselmoinnin puute sekä ylläpidon unohtaminen.

Kaikki ohjelmistotekniikan ongelmat eivät liity aikatauluun tai kustannuksiin, sillä ongelmia voi esiintyä myös ohjelman suorituskyvyssä tai ulkonäössä. Suorituskyvyn ongelmat on tiedostettu jo pitkän aikaa, ja tiedetään, että ne johtuvat puutteellisesta suunnittelusta ja etenkin koodauksesta. Sen sijaan ohjelman ulkonäkö ja helppokäyttöisyys on jäänyt vähemmälle huomiolle, vaikka etenkin kaupallisessa ohjelmassa käyttöliittymä on tärkeä kilpailutekijä, esim. Macintoshin menestys on suurelta osin käyttöliittymän ansiota. Markku Metsämäen mukaan graafisen käyttöliittymän suunnittelun ongelmia ovat tiedon puute graafisen suunnittelun peruskäytännöistä, tekstitiedon puute ja luovan ajattelun kahlitseminen toteutustasolla (2, s. 59). Mielestäni yksi käyttöliittymän ongelmista on myös se, että käyttöliittymän suunnittelussa ei tunneta, tai ei oteta huomioon ohjelman lopullista käyttäjää, joten suunnittelija tekee liittymästä "oman suun mukaisen".

Osa ohjelmiston kehityksen ongelmista johtuu vääristä uskomuksista ja tottumuksista, jotka kummittelevat johtajien, käyttäjien, ja mikä pahinta, myös ohjelmistoammattilaisten mielissä. Yleensä uskomuksissa on mukana syvällisiä viisauksia, mutta ohjelmistotekniikan alalla asia on valitettavasti päinvastoin. Ne vievät harhaan ja ovat hyvin vaarallisia, koska ne yleistävät asioita ja johtavat vääriin asenteisiin. Yleensä uskotaan, että yleisluonteinen tavoitteenasettelu riittää ohjelman kirjoittamiseksi, vaikka todellisuudessa se on ohjelmisto-

hankkeen epäonnistumisen pääsyy. Toinen yleinen uskomus on, että ohjelma on valmis, kun se on kirjoitettu ja saatu toimimaan. Ojalehdon mukaan todellisuudessa ohjelman elinkaari muodostuu esitutkimuksesta, määrittelystä, suunnittelusta, koodauksesta, testauksesta ja ylläpidosta - ohjelman koodaus on vain pieni osa (10 - 20 %) työstä ja kustannuksista. (1, s. 29 - 30.)

Kaikkiin edellä mainittuihin ongelmiin voidaan vaikuttaa oikealla ohjelmistotekniikalla - sen opettamisella ja noudattamisella. Sen tärkeys korostuu koko ajan, sillä ohjelmistojen osuus erilaisissa järjestelmissä kasvaa jatkuvasti. Ohjelmisto sopii kaikkeen, missä toimitaan etukäteen määritettävissä olevien sääntöjen mukaan (tekoälyjärjestelmät poisluettuina). Tyypillisiä ohjelmiston sovellusalueita ovat systeemiohjelmistot, automaatiosovellutukset, kaupallishallinnolliset sovellutukset, tekniikka ja tiede, tekoäly- ja asiantuntijajärjestelmät sekä sulautetut järjestelmät.

2.2 Ohjelmiston elinkaari ja kehitysmallit

Ohjelmiston elinkaari on periaatteessa samanlainen kuin minkä tahansa tuotteen elinkaari, eli se alkaa ohjelmiston käyttötarpeen toteamisesta, sisältää kehitysvaiheet, käyttö- ja huoltovaiheet, sekä päättyy tuotteen tekniseen vanhenemiseen ja käytöstä poistumiseen. Hyvin määritellyiden menetelmien ja vakiintuneiden ohjelmistoapuvälineiden käyttö yhdessä katselmuksien kanssa jokaisessa ohjelmistoelinkaaren kohdassa on Ojalehdon mukaan hyvän ohjelmistotekniikan peruste (1, s.17).

Ohjelman kehitys voidaan jakaa kolmeen vaiheeseen: määrittelyyn, suunnitteluun ja ylläpitoon. Määrittelyvaiheen tehtäviin kuuluvat esitutkimus ja vaatimusmäärittely. Esitutkimustyö aloitetaan ennen varsinaista ohjelmiston kehitystä. Esitutkimuksen tarkoituksena on selvittää voidaanko, kannattaako ja ehditäänkö aiottua hanketta tehdä. Vaatimusmäärittely on aikajärjestyksessä seuraava työvaihe ja sen tehtävänä on selvittää ja kuvata ohjelmiston toiminnot, tiedon kulku ja tietosisällöt, rajapinnat sekä suorituskykyvaatimukset ja -rajoitukset. (1, s. 17 - 18; 3, s. 1 - 3.)

Suunnitteluvaiheen tehtäviä ovat ohjelmiston arkkitehtuurin eli rakenteen suunnittelu, yksikkösuunnittelu (moduulisuunnittelu), koodaus ja testaus. Rakennesuunnittelussa suunnitellaan ohjelmiston rakenne eli jaetaan se moduuleihin. Moduulisuunnittelulla kuvataan ohjel-

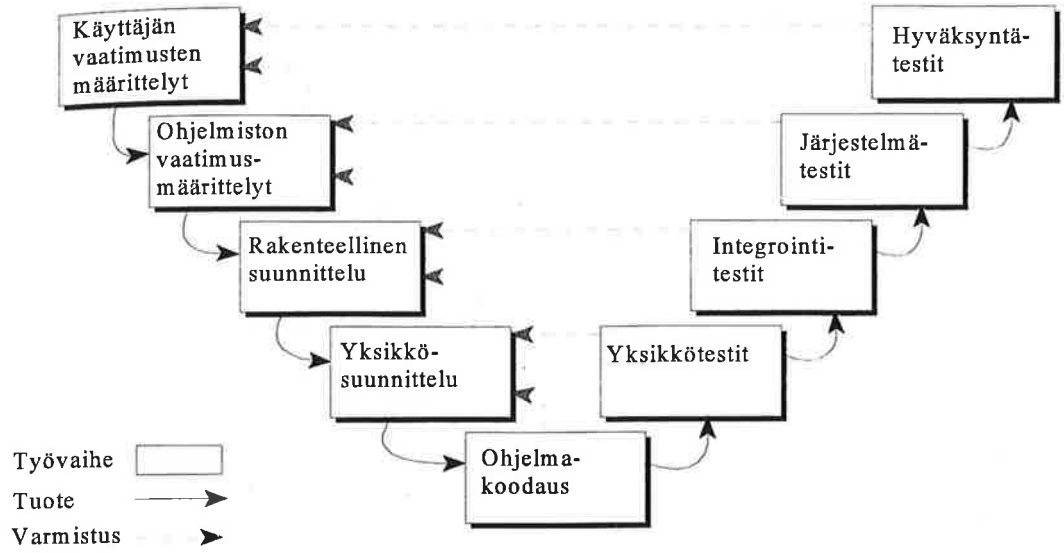
man toiminta lohkokaavioilla tai pseudokielellä. Koodaus on suoraviivainen työvaihe, jolla toteutetaan moduulien toiminta. Testaustyö jakaantuu neljään osaan. Ensin yksikkötestauksessa varmistetaan, että kukin moduuli sinänsä toimii oikein. Integrointitestauksessa moduulit kootaan rakennesuunnitelman mukaan paketiksi ja todetaan kokonaisuuden toimiminen. Järjestelmätesteillä osoitetaan, että ohjelmisto toimii ohjelmistovaatimusten mukaisesti. Kelpoisuustestissä katselmoidaan asiakkaan kanssa koko ohjelmisto vaatimusmäärittelyn perusteella. (1, s. 23 - 24; 3, s. 3.)

Ylläpidon tehtävän on yksinkertaisesti ylläpitää ohjelmistoa. Alun perin ylläpitoa ei otettu huomioon ohjelmistotyössä, mutta myöhemmin on opittu (kantapään kautta), että ylläpito vie 70 % kaikesta ohjelmankehityskapasiteetista. Ylläpidon merkitystä ohjelmistotyössä voidaan ehkäpä pitää tärkeimpänä. Ylläpidon tehtävät vaihtelevat huomattavasti riippuen sovelluksen ominaisuuksista, mutta yhteisiä piirteitä ovat asiakaspalautteen kerääminen ja tarkka dokumentointi. Ylläpidosta saadaan usein arvokasta käyttökokemustietoa tulevien projektien hyödyksi. (1, s. 24; 3, s. 3.)

Ohjelmiston yksinkertaisinta ohjelmakehitysmallia, mikä soveltuu pienehköille ohjelmistoille, voidaan kuvata ns. vesiputousmallilla, joka on esitetty kuvassa 1. Kehitys alkaa tuotteen käyttäjän vaatimusten määrittelyllä ja päättyy lopullisen ohjelmatuotteen hyväksyntätesteihin. Jokaisesta kehitysvaiheesta kootaan palaute edelliseen vaiheeseen, joka on oleellisen tärkeätä ohjelman laadun todennuksen (verifioinnin) kannalta. Kun kunkin työvaiheen tulee perustua edellisen vaiheen työlle, niin palautteen kokoamisella ja sen käsittelyllä varmistetaan, että jälkimmäisen vaiheen kehitystulos on aikaisemmin tehtyjen määrittelyjen mukainen ja uudet ideat tulevat samalla hyväksikäytetyiksi edellisen vaiheen työssä.

Kehitysmalli, josta voi käyttää nimitystä "vähittäinen tuotteen luovutus asiakkaalle" toimii siten, että ohjelman rakennesuunnittelu ja sitä seuraava suunnittelu johtaa jonkin ohjelman osan prototyyppiin, jonka asiakas hyväksyy koekäyttöön. Tästä saadut kokemukset siirretään välittömästi kehitystyön tekijöille ja jatketaan uudella rakennesuunnittelulla lopullista täydellistä versiota varten. Tämä menettely soveltuu tapauksiin, missä ohjelman osittainen käyttö on todella mahdollinen, koska prototyypin on oltava käyttökelpoinen osa ohjelmistoa. Haittapuolena tälle mallille saattaa olla se, että asiakas saa osan tuotetta jo etukäteen. Tällöin tarvitaan lisää testejä ja huolenpitoa, että lopullinen ja täydellinen versio, jonka kehitys siis oli luovutushetkellä kesken, ei millään tavoin tule poikkeamaan aikaisemmasta

luovutetun ohjelman osalta. (3, s. 4 - 5.)



KUVA 1 Ohjelmistokehityksen vaiheet (3, s. 4)

"Evoluutiomalli" poikkeaa edellisestä siinä, että asiakkaalle luovutettavat versiot seuraavat toisiaan lähes jokaisesta kehitysvaiheesta. Kaikki kehitysvaiheet käydään läpi ja toimintakuntoisia kehitysmalleja luovutetaan koekäyttöön välittömästi ohjelmiston tai sen osan valmistumisen jälkeen. Tällöin asiakas saa mukana vain vähimmäisohjeet ja lisäksi esitetään, mitä vaatimuksia toivotaan täydennettäväksi. Evoluutiomallin suurin haitta on se, että vaatimukset valmistuvat vasta kehitystyön kuluessa. Työ sisältää siten sen riskin, että myöhempi vaatimustaso saattaa vaatia alkuperäisen ohjelmarakenteen uusimista jossakin vaiheessa kokonaan, mikä johtaa helposti kustannusten laajenemiseen. Lisäksi tilapäisiksi tarkoitetut ratkaisut saattavat jäädä pysyviksi osiksi ja voivat siten vääristää projektin kokonaisuuden hallinnan. Toisaalta evoluutiomallin kiistaton etu muihin kehitysmalleihin on se, että käyttäjän toivomukset ovat aina saatavilla, eli käyttäjä käytännössä osallistuu kehitystyöhön projektiryhmän mukana. (3, s. 5.)

2.2.1 Käyttäjän vaatimusten määrittely

Käyttäjän vaatimusten määrittely on ohjelmiston kehityksessä ongelman määrittelyvaihe. Hyvä suunnittelu ja toteutus ei voi korvata vaatimusmäärittelyn puitteita, sillä ilman vaatimusmäärittelyä ei tiedetä mitä pitäisi tehdä. Käyttäjän vaatimukset eivät aina ole tarjolla sellaisinaan, sillä tuleva käyttäjä ei aina ole alan asiantuntija, eikä siten pysty esittämään täsmällisiä ja yksityiskohtaisia vaatimuksia kehittäjälle. Kehittäjän tehtävänä onkin toimia haastattelijana, konsulttina ja ongelmien ratkojana - kehittäjä ja käyttäjä tekevät jatkuvasti

tiivistä yhteistyötä. Olemassaolevat vastaavat ohjelmat tai niiden prototyypit ovat tärkeä tuki, kun on määriteltävä uuden rinnakkaisen tai suorituskyvyltään vastaavan tuotteen ominaisuudet ja vaatimukset.

Vaatimusten analysoinneissa toistuvat samat rutiinit. On määriteltävä käyttöympäristö, eli laitteistovaatimukset, toimintaolosuhteet eli vuorovaikutteisissa ohjelmissa käyttäjien tausta ja koulutus, sekä käytön tavoitteet, ohjelmiston ja laitteiston liittyminen muihin järjestelmiin jne. Käyttäjän vaatimukset voidaan jakaa karkeasti kahteen ryhmään: vaatimuksiin, jotka kohdistuvat ohjelmiston suorituskykyyn ratkaista käyttäjän ongelma, ja rajoituksiin, jotka johtuvat käyttäjän mahdollisuuksista ko. ongelman ratkaisuun ohjelmiston avulla.

Vaatimusanalyysi on vaikeaa ja vaativaa viestintätyötä, jossa virheitä ja väärinkäsityksiä syntyy helposti, tai jokin erityispiirre jää huomaamatta. Tämän vuoksi vaatimukset dokumentoidaan seuraavaa työvaihetta ja lopullista testausta varten. Vaatimusluetteloon voidaan jokaisen vaatimuksen kohdalle liittää tunnistenumero ja arvio siitä onko kyseessä oleellinen käyttäjän vaatimus (ei neuvotteluvaraa), sekä prioriteettiaste, eli kehitystyö voidaan jakaa sen perusteella tärkeysjärjestykseen. Lisäksi vaatimukseen voidaan liittää vaatimuksen antaja, etenkin jos vaatimuksia kootaan eri lähteistä.

2.2.2 Ohjelmistovaatimusten määrittely

Ohjelmistovaatimusten määrittely on Ojalehdon mukaan ohjelmiston kehityksessä ongelman analyysivaihe. Tässä suunnitteluvaiheessa analysoidaan edellisessä vaiheessa kootut vaatimukset ja kehitetään joukko ohjelmistomäärittelyjä niin täydellisinä, oikeina ja vaatimusten kanssa yhtäpitävinä kuin mahdollista. Jotta koko projektin epäonnistumisen riski saadaan minimoitua, on tässä kehitysvaiheessa oltava mukana kaikki käytettävissä oleva asiantuntemus ohjelmistosuunnittelijoilta, laitteistoasiantuntijoilta ja käyttöhenkilöstöiltä. (1, s. 93; 3, s. 6.)

Ohjelmistovaatimusten määrittelystä dokumentoidaan se, mitä ohjelma valmistuttuaan tekee, sekä ne määrittelyt, joita käyttäen on aikanaan valmistuva ohjelmisto ja sen dokumentaatio tarkastettava. Sen sijaan kuinka ohjelman suorituskyky toteutetaan, ei kuulu näihin dokumentteihin lukuun ottamatta mahdollisten rajoitusten kirjaamista (laitteisto, muistin koko jne.).

Ohjelmistovaatimusten kokoaminen etenee normaalisti siten, että edellisen vaiheen tuloksista muotoillaan loogisia kokonaisuuksia ja hahmotetaan eri ratkaisuvaihtoehtoja. Toisin sanoen käyttäjän vaatimukset muutetaan ohjelmanlaadinnan kannalta sopiviksi osatavoitteiksi. Kehitykseen kuuluu usein ohjelmaprototyyppien ja mallien laadinta, joilla selvennetään käyttäjän vaatimusten mahdollisesti aiheutuneita ongelmia. Ahlbergin mukaan etenkin Windows-sovellutuksissa mallintaminen sovellutuksen varhaisessa vaiheessa on hyvin suositeltavaa, koska se auttaa sekä käyttäjiä että toteuttajia ymmärtämään valmistuvaa sovellutusta (4, s. 188). Käyttöliittymät ulkoisiin laitteisiin ja käyttöliittymä on syytä luonnostella ja tarkistaa mahdollisimman aikaisessa vaiheessa. (3, s. 7 - 8.)

Vaatimusten pohjana käytetään ohjelmiston toteutuksesta riippumatonta loogista mallia, mikä rakentuu käyttäjän vaatimusten pohjalle. Mallin eri lohkot muodostavat selvityksistä "mitä on tehtävä" kussakin ohjelmalohkossa. Seuraavassa suunnitteluvaiheessa kehitetään tästä kysymykseen "miten se tehdään" vastaava fyysinen malli, joka erityisesti ottaa huomioon käyttöympäristön ja kaikki kehitystyökalut, mitkä ovat käytettävissä. Loogisen mallinnuksen periaatteena on "top-down"-suunnittelu, eli edetään käyttäjän päävaatimusten toteutuksesta asteittain alemmille detaljitasoille. Ennen siirtymistä seuraavalle detaljitasolle, on suoritettava eriasteisia läpikäyntejä, katselmuksia ja tarkastuksia. Siten voidaan varmuudella todeta, että malli täyttää vaatimukset. (1, s. 98; 3, s. 8.)

Looginen ja fyysinen mallinnus kuuluvat rakenteellisen analyysin mukaiseen mallirakenteeseen. Sen taustalla on Tom De Marcon rakenteellisen analyysin ja suunnittelun ja systeemisuunnittelun menetelmä (Structured Analysis and System Specification). Loogisella mallinnuksella kuvataan siis vaatimusmäärittelyn tulos (mitä ohjelma tekee) ja fyysisellä suunnittelun tulos (miten ohjelma sen tekee). Looginen malli muodostuu hierarkkisesti seuraavista kuvauksista:

-ympäristömalli (Enviromental Model)
    -rajapintakaavio (Context Diagram)
    -tapahtumalista (Event List)
-käyttäytymismalli (Behavioral Model)
    -tietovuokaavio (Data Flow Diagram)
    -tilakaavio (State Diagram)
    -er-kaavio (Entity Relationship Diagram)
    -tietohakemisto (Data Dictionary).

Tällaisen selvästi määritellyn mallintamisen etuna on se, että on sovittu tarkkaan, mitä ympäristö- tai käyttäytymismalli tarkoittaa ja millaisista kuvauksista se koostuu. Loogisessa ja fyysisessä mallinnuksessa käytetään hyväksi Yordonin RT/SA/SD-menetelmää (Real Time Structured Analysis and Design). RT/SA/SD on hyvin pitkälle graafisiin kuvioihin perustuva menetelmä, jota täydentävät lähinnä alimpien tasojen yksityiskohtien spesifioinnissa tekstimuotoiset dokumentit. Menetelmä on yleiskäyttöinen, eli sitä voidaan käyttää myös muiden järjestelmien kuvaamiseen, esim. yrityksen organisaatio. (3, s. 8 - 9.)

Ohjelmistovaatimukset voidaan esittää myös virtausmallina, jossa näkökulma on tietojen virtaamisessa. Virtausmallilla kuvataan informaation kulkua järjestelmän läpi ja siinä tapahtuvia muutoksia. Virtausmallinnuksessa käytetään yleensä tiedon virtauskaaviota (Data Flow Diagram), joka on graafinen ja erittäin ilmaisuvoimainen esitystapa. Virtauskaavioita käytetään paljon myös muissa ohjelmakehityksen vaiheissa, kuten esim. moduulien vuokaavioissa. Tietovuokaaviossa korostuu datavirta, ei niinkään toimintojen ohjaus. Tietovuokaavio kertoo siis tiedon kulun, ei esim. toimintaehtoja ja -silmukoita. Samoin kuin RT/SA/SD-menetelmässä systeemialkioita voidaan hienontaa tarkemmiksi kuvauksiksi, eli tietovuokaavio voidaan kerrostaa (top-down). Myös tietovuokaavio soveltuu lähes minkä tahansa systeemin kuvaamiseen. (3, s. 8 - 9.)

Ohjelmistovaatimukset voidaan luokitella seuraaviin alaluokkiin:

-toiminnalliset vaatimukset
-toiminnan mittaavat ominaisuudet, suorituskyky
-liityntävaatimukset
-käyttövaatimukset
-resurssivaatimukset
-verifiointivaatimukset
-hyväksyntätestien vaatimukset
-dokumentaation vaatimukset
-varmuusvaatimukset
-siirrettävyysvaatimukset
-laadun vaatimukset
-luotettavuusvaatimukset
-huollettavuus
-turvallisuusvaatimukset (3, s. 15 - 16).

Toiminnallisilla vaatimuksilla määritellään mitä ohjelmiston tulee tehdä, mikä on sen tarkoitus. Toimintaa mittaavilla ominaisuuksilla määritellään kapasiteettiarvoja, toimintanopeuksia jne., joilla on yhteys vaadittaviin toimintoihin. Liityntävaatimuksilla kerrotaan laitteisto-, ohjelmisto- ja kirjasto-ohjelmavaatimukset niiden elementtien kohdalta, joiden kanssa kehitettävän ohjelmiston on toimittava tai joita se käyttää hyödykseen. Käyttövaatimuksilla määritellään järjestelmän käyttöominaisuudet, ts. kuinka ohjelmiston käyttäjä toimii sen kanssa eri tilanteissa (esim. virhetilanteet, virheviestien muoto, aputekstit jne.). Resurssivaatimuksilla tarkoitetaan laskentatehon, muistitilan, levyasemien jne. ylimpien raja-arvojen määrittelyä. Verifiointivaatimuksilla määritellään valmistuneen ohjelmiston ominaisuuksien tarkistukseen liittyviä seikkoja, kuten esim. simulaatio-ohjelmien tai laitteiden käyttö. Hyväksyntätestien vaatimuksiin kuuluvat määrittelyt, joiden mukaan hyväksyntätestit suoritetaan, esim. standardit tai asiakkaan vakiomenettelyt. Dokumentaation vaatimuksia tarvitaan projektikohtaisen dokumentoinnin yksityiskohtaiseen määrittelyyn. Varmuusvaatimuksilla määritellään vaatimukset menetelmille, joilla estetään ohjelmiston asiaton käyttö, tietosuojan asettamat vaatimukset, salasanat sekä virustorjunnan keinot. Ohjelmiston tietosuojaan kuuluu myös fyysinen varmennus, esim varmuuskopioiden teko, levykkeiden säilytys palovarmassa tilassa jne. Siirrettävyysvaatimukset määrittää tietojärjestelmät, joissa ohjelmiston tulee toimia ilman muutoksia ja järjestelmät, joissa sen tulee toimia modifikaatioiden jälkeen. (3, s. 16 - 18.)

Ohjelmiston laadulle asetettavat vaatimukset perustuvat siihen, että on tunnettava tarkoin ohjelmiston tavoitteenmukainen käyttö ja sen (ja käyttäjän) asettamat vaatimukset. Ohjelmiston laatu määritellään luotettavuuden, huollettavuuden ja turvallisuuden osalta. Luotettavuus voidaan määritellä esim. keskimääräisen vikavälin (MTBF, Mean Times Between Failures) avulla. Huollettavuus vaatimuksia voi olla esim. ohjelmasta löytyvien virheiden korjaamismenetelmiin liittyvät määrittelyt. Turvallisuusvaatimuksiin sen sijaan kuuluvat määrittelyt, joiden avulla pyritään vähentämään vahingon mahdollisuutta, kun sen aiheuttajana on ohjelmistovika. (3, s. 17)

2.2.3 Rakenteellinen suunnittelu

Rakenteellinen suunnittelu on ohjelmiston kehityksessä ongelman ratkaisuvaihe. Aikaisemmin ohjelmiston toteutus alkoi jonkin ohjelmiston osan koodaamisesta, ja pala kerrallaan pyrittiin toimivaan lopputulokseen. Ongelma oli siinä, että ohjelmistoa ei nähty kokonaisuu-

tena. Ojalehdon mukaan rakennesuunnittelun eli alustavan suunnittelun tehtävänä on kehittää ohjelmisto ylhäältä alaspäin, kokonaisuudesta tieto- ja toimintosisällön mukaiseksi rakenteeksi, ohjelmistorakenteeksi eli -arkkitehtuuriksi, jota iteroidaan ja kehitellään ennen kuin mennään eteenpäin (1, s. 132). Rakennesuunnittelu perustuu määrittelyvaiheessa koottuun, ja tietovuokaavioina ja tietorakenteina esitettyyn informaatioon. Tämä tietous muunnetaan systemaattisin menetelmin ohjelmistorakenteeksi ja moduulien kuvauksiksi. Rakennesuunnittelu on siis ohjelmiston rakenteen jakamista moduuleihin, ja tarvittavien tietojen ja tietorakenteiden liittämistä niihin. Lisäksi rakennesuunnittelulla kuvataan moduulien välillä liikkuvat tiedot.

Rakenteellisen suunnittelun päätehtäviä ovat

-fyysisen mallin kehittäminen
-kokonaisrakenteen määritteleminen
-ohjelmointikielen valinta
-suunnittelun arviointi ja vaihtoehtojen karsinta (3, s. 20).

Suunnittelu siis etenee siten, että edellisen vaiheen loogisen mallin lohkot, joissa on esitetty "mitä on tehtävä" muunnetaan lohkoiksi, joissa esitetään "miten se tehdään". Fyysisen mallin kehitystyö vaatii kehityspäätöksiä, jotka johtavat toimintojen muuntoon ohjelmakomponenteiksi ja joiden syöttö- ja lähtösuureet määritellään. Suunnittelussa on otettava huomioon myös ei-toiminnalliset vaatimukset ja toteutustavan (työkaluohjelmien) asettamat vaatimukset.

Ohjelmiston jako moduuleihin (komponentteihin) voidaan tehdä esimerkiksi toiminnallisin perustein tai sen mukaan, kuinka ohjelma kommunikoi sisäisesti ja ulkoisesti. Tärkeintä on löytää joukko sisäisiä hierarkiatasoja eri lohkojen sijoittamiseksi määrätyille paikoilleen ohjelmarakenteessa. Ojalehto (1, s. 133) painottaa top-down-suunnittelua moduuleihin jaossa. Jaottelu mahdollistaa tällöin ohjelman hallinnan siten, että ylemmän tason suunnittelu voidaan tehdä valmiiksi ilman alemman tason detaljien käsittelyä samanaikaisesti - ainoastaan toiminnat ja liitynnät tulee tietää aluksi myös detaljitasolla.

Rakennesuunnittelulle on ominaista, että kuvauksissa ei vielä käsitellä tiedostoja, tietueita tai tavuja. Sen sijaan määritellään eri lohkojen vaatimusten mukaisia tehtäviä sekä niiden välinen tietoliikenne. Rakenteellisen suunnittelun alin taso käsittää ohjelmakomponentteja, jotka ovat melko itsenäisesti suunniteltavasti myöhemmin tai otettavissa valmiina ohjelma-

kirjastoista.

Rakenteellisen suunnittelun tuloksena syntyy fyysisen mallin (tai muun vastaavan mallin) dokumentit. Tuloksen tulee käsittää (lohko)kaaviot kullekin rakennetasolle, mistä ilmenevät eri tietovirrat ja ohjaustoiminnat ohjelmiston komponenttien välillä. Ohjelmiston tarvitsemat tiedostot, eri komponenttien tehtäväkuvaukset, rajoittavat kriteerit sekä komponenttien liitäntöjen ja toiminnan kuvaukset muodostavat oleellisen osan tuloksesta. Tärkeätä on esittää toiminnan kuvaus siitä, mitä vaaditaan, eikä niinkään sitä, kuinka ko. toiminnan vaatimus toteutetaan.

2.2.4 Yksikkösuunnittelu, koodaus ja läpikäynnit

Ratkaisun toteutusvaihe käsittää yksikkösuunnittelun, ohjelmiston koodaukset ja lisäksi testaukset. Toimenpiteitä ovat siis:
-ohjelmayksikköjen eli -komponenttien (moduulien) suunnittelu
-koodaus
-ohjelmiston kokoonpano, eli integrointi
-koodattujen ohjelmayksikköjen testaus.

Toteutusvaiheen perustana ovat edellisen suunnitteluvaiheen dokumentit. Yksikkösuunnittelussa ohjelmistokokonaisuus kehitetään hajoittamalla määrittelyvaiheessa kuvattuja ja rakenteellisen suunnitteluvaiheen tuottamia lohkoja alemmille tasoille yhä tarkempiin ja tarkempiin komponentteihin ja niiden kuvauksiin (top-down), kunnes päästään ohjelmiston alimmalle tasolle.

Rakenteellinen (strukturoitu) ohjelma koostuu kolmesta perusalkiosta: peräkkäisyys, ehto ja toisto. Ojalehdon mukaan ohjelma voidaan aina koota näistä rakenteista ja ohjelmalla tulee olla vain yksi tulo- ja lähtöpiste (1, s. 199). Yksikkösuunnittelun ratkaisun esitystapoja ovat graafiset esitykset, taulukot ja pseudokieliset kuvaukset. Tunnetuimpia ja eniten käytettyjä esityksiä ovat graafiset vuo- eli lohkokaaviot ja laatikkokaaviot, joiden rakennusalkioita ovat peräkkäisyys, ehto ja toisto. Muita käytettyjä esityksiä ovat päätöstaulut, joilla kuvataan ohjelmistokomponentin päätösehtoja ja valintatilanteita, sekä pseudokielet, joilla voidaan kuvata mm. muuttujamäärittelyt, lohko-, ehto- ja toistorakenteet, aliohjelmamäärittelyt, liitäntäkuvaukset ja I/O-rakenteet.

Ohjelmistokomponenteille suunnitellaan yksityiskohtaiset rakennekuvaukset ja täten valmistellaan koodausvaihe niin pitkälle kuin mahdollista askel askeleelta. Tavoitteena on saada koodi toimivaksi jo ensimmäisellä kerralla. Oleellista tietenkin on se, että yksiköiden toiminta ja liitynnät ovat täysin selvitetyt ennen koodausvaiheen aloitusta. Vaikka suunnittelun tulisi edetä ylhäältä alas, jotkut alemman tason tai uudet komponentit on mahdollisesti suunniteltava ja koodattava ensin (esim. laiteohjaimet ja käyttäjäkirjastot).

Koodauksessa muunnetaan moduulisuunnitelma jollekin ohjelmointikielelle. Siitä eteenpäin muunnosketju jatkuu automaattisilla välineillä (kääntäjillä, linkittäjillä) suorituskelpoiseksi ohjelmaksi, jota voidaan testata tietokoneella. Koodauksen edetessä myös suunnitteluoletusten, toimintojen, rakenteen, liitäntöjen, sisäisten tietorakenteiden kehityksen ja resurssien käytön dokumentoinnin tulisi edeltä samanaikaisesti. Suunnittelutiedon tulisi aina olla mukana tiivistettyinä ja selkeinä kommentteina lähdekielisessä koodissa.

Integroinnilla tarkoitetaan ohjelmistokomponenttien (moduulien) yhdistämistä toisiinsa toiminto kerrallaan, "ylhäältä alaspäin" menettelyn mukaisesti. Integroinnissa käytetään apuna lyhyitä "koodintynkiä" esittämään puuttuvia alemman tason moduuleita, kunnes ne saadaan koodattua ja testattua. Integroinnin eri menetelmillä on yhtenä päätavoitteena lyhentää ja helpottaa testausta.

2.2.5 Testaus

Testit ovat suunnitelman verifiointivaihe, jolla etsitään ohjelmasta virheitä joko manuaalisesti tai tietokoneen avulla. Oletuksena voidaan hyvin pitää, että ohjelmassa on aina virheitä. Testauksella varmistetaan, että ohjelma toimii asetettujen vaatimusten mukaisesti, joten testauksella on keskeinen osa ohjelmiston laadun varmistamisessa. Testaukseen tarvittava aika arvioidaan yleisesti liian pieneksi. Ohjelmiston kehityskustannukset tulevat sitä suuremmiksi mitä myöhäisemmässä vaiheessa virheet löydetään. Kustannusten alentamiseksi on virheiden etsimiseen paneuduttava kaikissa suunnittelun vaiheissa.

Ohjelmiston testaus jaetaan usein neljään eri vaiheeseen: yksikkötestaus, integrointitestaus, järjestelmätestaus ja hyväksymistestaus. Yksikkötestauksessa ohjelmisto testataan pieninä osina. Testattava moduuli on aliohjelma, rajoitettu aliohjelma, keskeytyspalveluohjelma jne. Yksikkötestauksen etuina ohjelmiston testauksen kannalta ovat, että koko ohjelmiston

testaus voidaan hallita paremmin ja virheiden löytäminen yksiköistä on helppoa. Myös yksikköjen samanaikainen testaus on yksikkötestauksessa mahdollista. Integrointitestauksessa testatut moduulit yhdistetään ja niiden yhteistoiminta tarkastetaan. Myös moduulien väliset liitännät ja ulkoiset liitännät testataan. Testausjärjestys voi olla joko ylhäältä alaspäin tai alhaalta ylöspäin, riippuen testattavan osakokonaisuuden tärkeydestä. Integrointitestaus huipentuu lopulta koko järjestelmän testaukseen. Järjestelmätestaukseen kuuluu koko ohjelmistojen testaus joko kehitysympäristölaitteessa tai kohdelaitteessa. Testauksella varmistetaan, toimiiko koko ohjelmisto haluttujen vaatimusten mukaisesti. (1, s. 239 - 250; 3, s. 30 - 33.)

Lopullinen testaus eli hyväksymistestaus tapahtuu yhdessä asiakkaan kanssa. Tavoitteena on osoittaa, että tuote toimii asiakkaan vaatimusten mukaisesti. Hyväksymiskriteerit on oltava dokumentoituna jo vaatimusmäärittelyn yhteydessä asiakkaan toimesta. Jokaisen hyväksymistestitapahtuman tuloksena on kaksi vaihtoehtoa: tulos on vaatimusten mukainen ja hyväksytään tai havaitaan poikkeama ja kirjoitetaan vikalista. Projektin tässä vaiheessa vikojen löytyminen merkitsee yleensä myös neuvotteluja aikataulumuutoksista. (1, s. 253)

2.2.6 Katselmukset

Aina, kun jokin vaihetuote on tekijän mielestä valmis, se voidaan katselmoida. Voidaan sanoa, että kehitystyön jokin osa on valmis, vasta kun se on katselmuksessa hyväksytty. Katselmus voidaan jakaa kahteen osaan, todentamiseen ja kelpoistamiseen. Todentamisella Ojalehdon mukaan todetaan vaatimusmäärittelyssä sovittujen hyväksymiskriteerien perusteella, mitkä osat määrällisesti tähän vaiheeseen kuuluvista asioista on tehty, sekä onko asiat johdettu oikein edellisestä vaiheesta (1, s. 25). Voidaan sanoa, että todentamisella tutkitaan onko tehty oikeaa asiaa. Kelpoistamisella Ojalehdon mukaan tutkitaan edellisen vaiheen läpäisseistä asiakirjoista, mitkä niistä täyttävät laadulliset kriteerit, ovatko esim. sisältö ja tarkkuus riittävät (1, s. 25). Kelpoistamisella siis todetaan, onko oikeat asiat tehty oikein.

2.3 Katsaus olio-ohjelmointiin

Vaikkakin rakenteellisen ohjelmoinnin (esim. C- ja Pascal-kieli) periaatteet helpottivatkin ohjelmien selkeyttä, luotettavuutta ja ylläpitoa, suuren ohjelman ohjelmointi on yhä haaste. Sovellutukset ovat muuttuneet laitteistokehityksen mukana yhä monimutkaisemmiksi ja

niiden koko on kasvanut. Siksi niiden ylläpitokustannukset ovat kasvaneet valtavasti. Suurin osa ylläpitokustannuksista aiheutuu Ahlbergin mukaan muutosten tekemisestä (4, s. 19). Ylläpitokulujen vähentämiseksi tarvitaan siis uusia tekniikoita, joiden avulla ohjelmaan tehtävät korjaukset ja muutokset ovat mahdollisimman pieniä. Perinteistä, eli osittavaa ohjelmistokehitystä käyttäen pienetkin muutokset ohjelman rakenteessa saattavat vaatia suuria muutoksia koko ohjelmistossa. Olioperustainen ohjelmointitapa antaa Ahlbergin mukaan uuden näkökulman tämän ongelman poistamiselle, sillä oliopohjainen rakenne pitää huolen siitä, että muutokset ovat mahdollisimman paikallisia (4, s. 19).

Olioperustainen ohjelmointitapa säästää kustannuksia myös ohjelmiston rakennusvaiheessa, sillä se helpottaa uudelleen käytettävän koodin luontia. Käyttämällä aikaisemmin laadittuja ohjelmistokomponentteja voidaan saavuttaa merkittäviä säästöjä. Tämä on erityisen tärkeää Windowsin kaltaisessa ympäristössä, jossa pelkästään käyttöliittymäelementtien hallinta ja kommunikointi ympäristön kanssa vaativat tuhansia koodirivejä. Oliopohjaisen ohjelmisto-kehityksen rakenteet ja niihin liittyvät mekanismit, kuten periytyminen ja monimuotoisuus, tarjoavat tehokkaat välineet ohjelmistokomponenttien uudelleenkäyttöön. Windows-luokka-kirjastot ja useimmat kehittimet sisältävät valmiita luokkia tällaisten rutiinitoimintojen käsittelyyn.

Oliosuuntautunut ohjelmointi perustuu 1960-luvulta peräisin olevan Simula-ohjelmointikie-len ideoille. Suurimman tutkimustyön olio-ohjelmoinnin alalta teki kuitenkin Xerox-yhty-män Palo Alto Research Center (PARC), joka kehitti kymmenen vuoden työn tuloksena ensimmäisen todella oliosuuntautuneen ohjelmointikielen, Smalltalk:in. PARC luultavasti tunnetaan parhaiten kehittämästään "Xerox Star"-mikrotietokoneesta ja sen graafisesta kuvakepohjaisesta käyttöliittymästä, joka inspiroi myös Apple Macintoshin ja Microsoft Windowsin kehittäjiä. Becksin mukaan tällaisen käyttöliittymän luomisen vaikeus oli syynä oliosuuntautuneiden ohjelmointitekniikoiden kehittämiseen. (6, s. 207.)

Ahlbergin mukaan oliopohjaisen ohjelmistokehityksen idea on lyhyesti siinä, että ohjelmas-sa pyritään mallintamaan todellisen maailman ilmiöitä. Tällä tavalla muodostetut kom-ponentit, oliot, ovat itsenäisiä ja suojattuja kokonaisuuksia. Niillä on tietty käyttäytyminen ja tiettyjä ominaisuuksia. Niitä voidaan muuttaa, lisätä ja poistaa vaurioittamatta koko ohjel-miston rakennetta. (4, s. 20.)

Tarkoituksenani ei ole tässä työssä perehtyä syvällisesti olio-ohjelmoinnin hienouksiin ja yksityiskohtiin, mutta ehkä on selventävää tarkentaa muutamia olioiden perusominaisuuksia.

Olio (object)

Pesosen ja Ihmeen mukaan olio on abstraktio jostakin ongelma-avaruuden kokonaisuudesta, jolla on tila, käyttäytyminen ja tunniste, jonka perusteella sen voi tunnistaa yksikäsitteisesti. Tila kuvaa olion sekä staattisia että dynaamisia tilaominaisuuksia. Becksin mukaan oliot muistuttavat ulkonaisesti tietueita, mutta eroavat siinä, että ne sisältävät omia proseduureja ja funktioita, joita molempia kutsutaan oliotyypissä metodeiksi (method). (5, s. 25; 6, s. 208.)

Luokka (class)

Ihmeen ja Pesosen mukaan samankaltaisten olioiden yhteneväisyydet, siis olion rakenne, toiminnot ja käyttäytyminen, voidaan kuvata luokan avulla (5, s. 25).

Ilmentymä (instance)

Meyerin mukaan luokan ajoaikaista ilmentymää, esiintymää, kutsutaan olioksi (5, s. 25).

Periytyminen (inheritance)

Becks määrittää perinnän siten, että olion ominaisuudet siirtyvät sen jälkeläisille. Periytymisen avulla luokkien ominaisuuksia ja toimintojan voidaan käyttää toisissa luokissa. Pesosen ja Ihmeen mukaan luokkien määrittelyssä voidaan käyttää hierarkiaa siten, että määritellään ensin yleinen luokka, yläluokka (superclass), jonka attribuutit ja operaatiot periytyvät hierarkiassa alempana olevalle luokalle, aliluokalle (subclass). (6, s. 209; 5, s. 25.)

Moniperiytyminen (multiple inheritance)

Moniperiytymisen avulla luokka voi periä ominaisuuksia usealta eri yläluokalta (5, s. 25).

Operaatio (operation)

Operaatio on luokan sisältämä palvelu (6, s. 25).

Metodi (method)

Metodi on luokan operaation toteutus. Metodi liittyy siis olioon.

Attribuutti (attribute)

Attribuutti on luokan sisältämä tieto (data).

Arvo (value)

Arvo on olion sisältämän tiedon arvo.

Polymorfismi (polymorphism)

Becksin mukaan polymorfismi eli monimuotoisuus on olion kyky muuntautua suorituksen aikana eri olotiloihin. Samanniminen operaatio esiintyy siis useassa eri luokassa. Polymorfinen olio voi ajon aikana tilanteen mukaan käyttää omia tai jälkeläistensä metodeja. Tällöin puhutaan virtuaalisista (virtual) metodeista. (6, s. 209)

Linkit, assosiaatiot (links, associations)

Pesonen ja Ihme määrittelee linkin seuraavasti. Linkki on fyysinen tai käsitteellinen liityntä olioiden välillä, esimerkiksi Ville Virtanen *työskentelee* M-kaupassa. Assosiaatio on liityntä luokkien välillä, esimerkiksi henkilö *työskentelee* yhtiössä. Linkki on assosiaation ilmentymä. (5, s. 26.)

2.3.1 Yleisimmät Windows-ohjelmointivälineet

Windows-ohjelmointiin on nykyään tarjolla valtava määrä erilaisia välineitä. Ahlberg jakaa välineet kahteen ryhmään: C/C++ välineisiin ja kehittimiin. Kehittimillä Ahlberg tarkoittaa sellaisia tuotteita kuten Visual Basic, Delphi, Paradox for Windows, PowerBuilder ja SQL-Windows. (4, s. 13.)

C/C++-kääntäjäpaketteja on lukuisia, mutta merkittävämmät tuotteet tällä hetkellä ovat

Borland C++ ja Microsoftin Visual C++. Molemmat tuotteet mahdollistavat ohjelmoinnin sekä C- että C++-kielellä. C++-ohjelmointia varten kummankin tuotteen mukana toimitetaan oma luokkakirjasto, joka pitää sisällään kaikki Windowsin ikkunoihin, sanomiin, kontrolleihin ja grafiikkaan liittyvät rutiinit ja täyttävät näin tehtävänsä. Ohjelmointikielten lisäksi molemmat kääntäjät tarjoavat lukuisia tarpeellisia työkaluja ohjelmakehitykseen, kuten resurssieditorin ja virheenkorjausohjelman. Microsoftin AppWizard ja Borlandin ClassExpert ovat apuohjelmia, jotka auttavat luomaan nopeasti uusien ohjelmien runkoja. Kummatkin tuotteet ovat ominaisuuksiltaan korkeatasoisia ja eroja etenkin suorituskyvyssä on vaikea löytää. Microsoftin ehdottomana etuna voidaan kuitenkin pitää reagointinopeutta uusiin käyttöjärjestelmän ominaisuuksiin.

Borland ja Microsoft taistelevat myös kehitin-ympäristössä keskenään. Microsoftin Visual Basic ja Borlandin Delphi ovat jopa ulkonäöltään hyvin samantapaisia kehittimiä, mutta eroavaisuuksia kuitenkin löytyy. Delphin ohjelmointikielenä on olio-Pascal ja siinä on oliotekniikalla toteutettu Pascal-kielinen Windows-luokkakirjasto, johon kehittimen eri työkalut tukeutuvat. Microsoftin Visual Basic sen sijaan käyttää nimensä mukaisesti ohjelmointikielenään Basic-kieltä, mutta ei tarjoa luokkakirjastoa, vaan tarvitsee käännetyn ohjelman tueksi erillistä tulkkia toimiakseen. Delphin kiistaton etu Visual Basiciin nähden onkin juuri sovellusten nopeus, joka saavutetaan konekielikääntäjän avulla - ominaisuus, jota Visual Basicin käyttäjät ovat pitkään kaivanneet. Delphillä voi tehdä myös DLL-kirjastoja (Dynamic Link Libraries), mutta suurin etu kuitenkin on sen oliosuuntautuneisuus. Delphillä luodut komponentit ja ohjelmaluokat ovat helposti uudelleenkäytettävissä.

Kehittimien yhteinen piirre on siinä, että sovellutukset "maalataan" näytölle. Sovellus kehitetään kahdessa vaiheessa: ensin valitaan komponenttipaletilta oman sovelluksen tarvitsemat kontrollit ja sen jälkeen kirjoitetaan sovelluksen logiikat kontrollien tapahtumakäsittelijöihin.

Valinta C/C++:n ja jonkin kehittimen välillä on periaatteessa hyvin yksinkertainen. C/C++-ohjelmointi vaatii enemmän aikaa ja rahaa, mutta vastapainoksi se tarjoaa nopeutta ja riippumattomuutta. Kehittimillä tuotettujen ohjelmien siirrettävyys muille käyttöjärjestelmille tai tietokannoille on usein hankalaa, jopa mahdotonta. Riskinä voidaan siis pitää sitoutumista kehittimen valmistajaan. Kehittimillä ohjelma syntyy kuitenkin huomattavasti nopeammin kuin C/C++:lla ja on yleensä helpommin ylläpidettävissä.

Valinta C:n ja C++:n välillä on myös hyvin yksinkertainen. C++ soveltuu paremmin isoihin sovellutuksiin kuin C, joka puolestaan soveltuu parhaiten pieniin ja tehokkuutta vaativiin ohjelmiin (esim. laiteohjaimet) sekä DLL-ohjelmointiin.

2.3.2 Olio-ohjelman suunnittelumenetelmät

Olio-ohjelmointi vaatii tuekseen myös uusia suunnittelu- ja analyysimenetelmiä. Ahlbergin mukaan tällä hetkellä suosituimpia ovat Rumbaughin OMT (Object Modeling Technique)- ja Jacobsonin OOSE-menetelmät. Niiden avulla voidaan hallita lähes koko ohjelman tuotantokaari: toimintaympäristön analysoinnista ohjelman suunnitteluun. Pesosen ja Ihmeen mukaan muita menetelmiä ovat mm. Booch, Buhr, Colbert, HOOD, Shlaer-Mellor, Kurz ja Odell. Jos suunnittelun tukena käytetään erityisiä CASE-välineitä (Computer Aided Software Engineering), voidaan suuri osa ohjelmakoodia tuottaa automaattisesti. (4, s. 43 - 44; 5, s. 14.)

Windows-ohjelmoinnissa olio-ohjelmoinnin merkitys kasvaa jatkuvasti. Ohjelmistoteollisuus on havainnut sen tarjoamat edut ja kaikki merkittävät sovelluskehitysvälineiden valmistajat ovat tuoneet markkinoille omat olio-kehitysympäristönsä.

3 FITWARE-JÄRJESTELMÄN MÄÄRITTELY

FitWare-järjestelmä koostuu viidestä osasta: FitWare-ohjelmasta, tietokoneesta, polkupyöräergometristä, sykemittarista ja purkulaitteesta, joka toimii tietokoneen ja sykemittarin välillä. Jäjestelmän tavoitteena on tarkentaa ja helpottaa ihmisen kestävyyskunnon analysointia. Järjestelmällä voidaan myös analysoida työpäivän ja harjoituksen fyysistä kuormittavuutta. Järjestelmän pääkohderyhmiä ovat mm. työterveyslaitokset, sairaalat ja kuntoutuslaitokset.

3.1 FitWare-järjestelmän teoreettinen tausta

Ennen FitWare-järjestelmän tarkempaa tarkastelua, on paikallaan selvittää ilmiöitä ja käsitteitä mihin järjestelmä liittyy. Olen koonnut tähän kappaleeseen lyhyen yhteenvedon FitWare-järjestelmän asiasisällöstä. Lähteinä näihin tietoihin on ollut henkilökohtaiset tiedonannot liikuntalääketieteen erikoislääkäri Hannu Litmaselta sekä fysiatrian erikoislääkäri Jukka Siljanderilta kevään 1997 aikana.

Liikuntaa säännöllisesti harrastavilla on muita pienempi vaara sairastua moniin verenkierto-elimistön, aineenvaihdunnan ja tuki- ja liikuntaelimistön sairauksiin. Lisäksi liikunta on tänä päivänä keskeinen osa-alue useimpien sairauksien kuntoutuksessa. Liikunnan terveyshyödyt välittyvät sekä biologisten että psykososiaalisten vaikutusmekanismien välityksellä, joista jälkimmäiset tunnetaan huonommin lähinnä mittaamiseen liittyvien ongelmien vuoksi.

Kaikkein selvimmin liikunnan terveysvaikutukset liittyvät kestävyyskuntoon (aerobiseen kapasiteettiin) eli kykyyn tuottaa hengitysilman hapen avulla energiaa pitkäkestoista lihastyötä varten. Aerobisen kapasiteetin paras yksittäinen mittari on maksimaalinen hapenkulutus (VO_{2max}). Riittävä maksimihapenkulutuksen taso on edellytys selviytymiselle arkipäivän kuormituksista. Kun elimistön ei tarvitse työskennellä lähellä suorituskyvyn ylärajaa, voidaan päivittäiset toimet suorittaa ylettömästi rasittumatta. Useimmat verenkierto- ja hengityselimistön sairaudet alentavat maksimaalista hapenkulutusta. Perintötekijöiden säätelemissä rajoissa pitkäkestoisella suuria lihasryhmiä kuormittavalla harjoittelulla jokainen voi parantaa aerobista kestävyyttään. Terveysliikunnan optimaaliseksi tehoksi kestävyyden osalta on määritelty 60 - 80 % kunkin yksilön maksimaalisesta hapenkulutuksesta (VO_{2max}). Liikuntaohjelmia laadittaessa määritellään tehon lisäksi harjoituksen kesto, useus ja liikuntamuoto.

Maksimaalista hapenkulutusta voidaan mitata suoraan uloshengitysilmasta ergospirometrialaitteistolla tai sitä voidaan arvioida epäsuorasti tavallisimmin sydämen sykintätason noususta kuormitusta lisättäessä. Kuormitusvälineenä käytetään yleisimmin polkupyöräergometria. Epäsuorat VO_{2max}:n mittaustavat eivät vaadi yhtä kalliita mittalaitteistoja kuin suorat mittaukset ja niille on tyypillistä submaksimaalisuus eli testattavaa ei tavallisesti kuormiteta uupumukseen saakka. Epäsuorat testit soveltuvatkin suurimmalle osalle väestöstä. Yleisimmin käytettyjä epäsuoria testejä ovat 12 minuutin kolmiportainen polkupyöräergometritesti sekä erilaiset askellus(step)testit. Epäsuorien testien joukkoon luetaan myös 2 km:n kävelytesti.

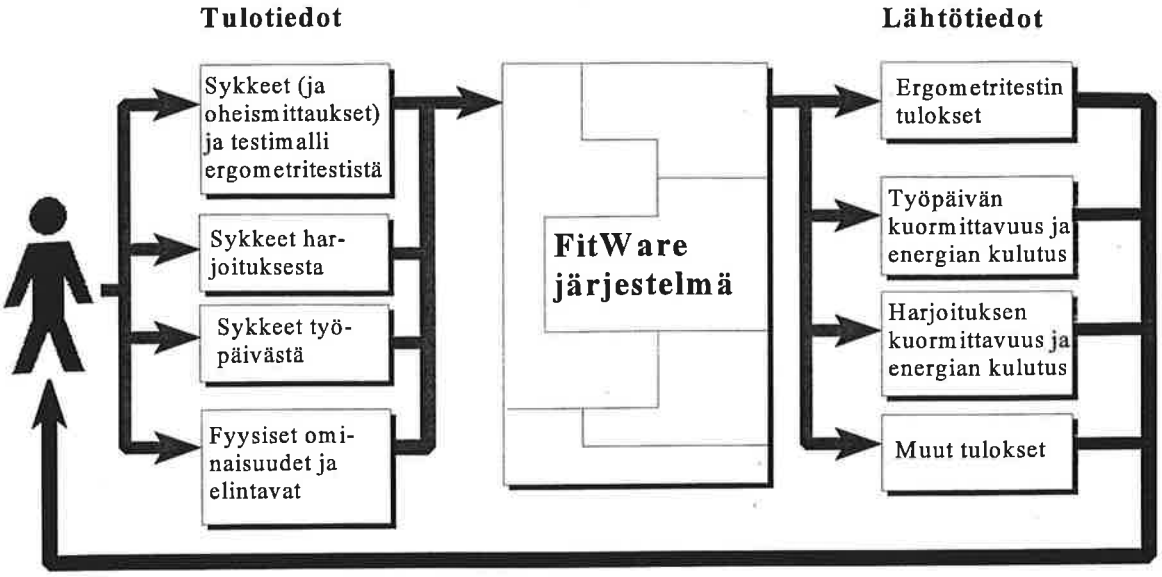
FitWare-ohjelma arvioi maksimaalista hapenkulutusta epäsuorasti polkupyöräergometritestissä, jossa kuormitusta nostetaan tasaisesti yleensä kerran minuutissa tai joka toinen minuutti 10 - 30 wattia kerrallaan testattavan koon ja kunnon mukaan kunnes tavoitesykintätaso (85 % maksimisykinnästä) on saavutettu. Kuormitusta vastaava sykintätieto kerätään muistilla varustettuun sykemittariin ja puretaan testin jälkeen tietokoneelle FitWare-ohjel-

maan. Sykintäkäyrän lineaarisesti nouseva osa ekstrapoloidaan maksimisykintätasolle, jota vastaava hapenkulutus arvioidaan hyvin dokumentoidulla jo 1950-luvulla esitetyllä tavalla. Verrattuna kolmiportaiseen 12 minuutin ergometritestiin kolmen kuorma-sykintäpisteen sijasta saadaan nyt n. 150 pistettä, mikä olennaisesti vähentää mittausvirheiden mahdollisuutta ja parantaa testin toistettavuutta. Lisäksi kuormituksen vähittäinen nostotapa on testattavalle miellyttävämpää.

Liikunnan ja erityisesti kestävyysliikunnan kansanterveydellinen ja taloudellinen merkitys on suuri. Kaikki liikunnan määrää väestössä lisäävät toimenpiteet edistävät myös terveyden määrää. Suorituskyvyn mittaaminen on perusta järkevälle liikuntaohjelmalle, jonka jatkuvuus ja seuranta ovat edellytykset pysyvien liikuntatottumusten syntymiselle.

3.2 Toiminnallinen kuvaus

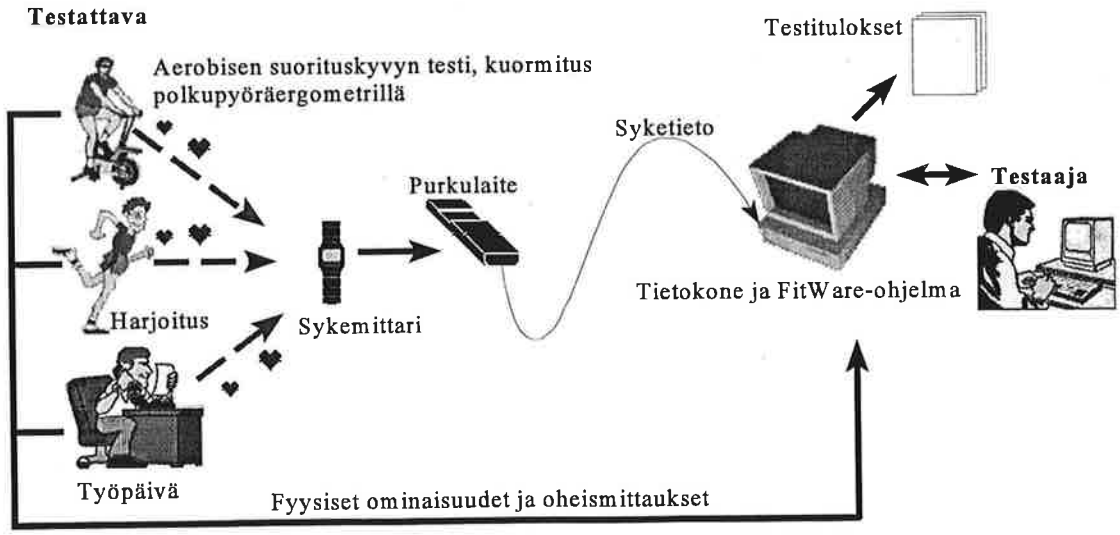
Kuinka FitWare-järjestelmä toimii ja mitkä ovat sen yksiköiden tehtävät järjestelmässä? Kuvassa 2 on kuvattu mitkä ovat järjestelmän tulo- ja lähtöarvot. Henkilön fyysiset ominaisuudet, kuten ikä ja paino, sekä syketiedot halutusta analyysistä määrävät tuloksen. Aerobisen suorituskyvyn testissä tarvitaan lisäksi tietää, mitä testimallia on käytetty. Lisätietoina järjestelmään voidaan syöttää oheismittaustuloksia ergometritestin ajalta, jotka ovat verenpaine-, PEF- ja BORG-mittaukset. Näitä tuloksia ei oteta huomioon tulosten laskennassa, mutta ne auttavat testaajaa henkilön analysoinnissa. Aerobisen suorituskyvyn tulokset saadaan laskettua ilman, että harjoituksen tai työpäivän analyyseistä saatavat tulokset olisivat selvillä. Sen sijaan harjoituksen ja työpäivän fyysistä kuormittavuutta ei voida määrittää, jos ei tiedetä aerobista suorituskykyä. Lähtötietoina FitWare-järjestelmästä saadaan tulokset ergometritestistä sekä kuormittavuudet ja energian kulutus työpäivästä ja harjoituksesta. Tuloksien perusteella testattavalle voidaan antaa liikuntaohjeita, joita noudattamalla hän voi parantaa suorituskykyään. Järjestelmän ainoana tarkoituksena ei ole tarkentaa ja helpottaa tulosten laskentaa, vaan myös ohjata ja motivoida sekä testaajaa että testattavaa.



KUVA 2 FitWare-järjestelmän toiminta

3.3 Järjestelmän osat

FitWare-järjestelmä koostuu FitWare-ohjelmasta, tietokoneesta, polkupyöräergometristä, sykemittarista ja sykepurkulaitteesta. Kuvassa 3 on esitetty eri komponenttien tehtävät FitWare-järjestelmässä. Sykkeet siirtyvät sykemittarilta purkulaitteen kautta FitWare-ohjelmaan. Ihmisen fyysiset ominaisuudet ja muut mittaustulokset käyttäjä syöttää suoraan ohjelmaan, jossa tapahtuu tulosten laskenta ja analysointi käyttäjän (testaajan) ohjauksella.

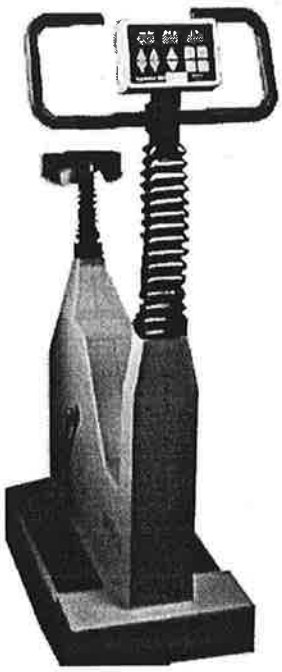


KUVA 3 FitWare-järjestelmän osat

Polkupyöräergometri toimii FitWare-järjestelmässä testauslaitteena, jolla kuormitetaan testattavaa henkilöä säätämällä poljettavaa vastusta. Vastusta säädetään sähköisesti, jotta se

olisi mahdollisimman tarkka. Mitä tarkempi polkupyöräergometri on, sitä tarkempi on testitulos. Polkupyöräergometrien kuormitusväli on yleensä 5-600 wattia ja pienin muutosväli 5 W. Ergometrin kuormittavuutta muutetaan ohjauspaneelissa olevilla mekaanisilla tai elektronisilla kytkimillä, mutta uudemmat ja kalleimmat ergometrit tukevat myös etäkäyttöä joko analogisella tai digitaalisella ohjauksella. Kuvassa 4 on esitetty Ergoline:n er800s - polkupyöräergometri.

Sykemittaria käytetään sykkeiden rekisteröintiin poljettaessa ergometriä. Myös harjoituksen ja työpäivän ajalta sykkeiden rekisteröiminen olisi mahdotonta ilman sykemittaria. Koska oman sykemittarin kehittäminen järjestelmään olisi ollut työnantajalle taloudellisesti mahdotonta, käytettiin järjestelmän kokoamisessa markinnoilla olevia sykemittareita. Tällä hetkellä ainoa sykemittarivalmistaja, jonka sykemittarit voidaan liittää FitWare-järjestelmään, on suomalainen Polar Electro Oy. Sykemittari koostuu anturista ja vastaanottimesta. Sykeanturi, jota kutsutaan myös sykevyöksi, kiinnitetään rintakehään ja vastaanotin, joka muistuttaa ulkonältään tavallista rannekelloa, kiinnitetään ranteeseen. Sykeanturi siirtää sähköiset sykäykset telemetrisesti vastaanottimelle - fyysisiä kaapeleita ei tarvita. Sykkeitä voidaan tallentaa kelloon eri tallennusväleillä, esim. 5:n, 15:n ja 60:n sekunnin välein. Pienempi tallennusväli antaa tarkemman tuloksen analysointivaiheessa, mutta kuluttaa enemmän kellon omaa tallennusmuistia. Kuvassa 5 on esitelty Polar Electro Oy:n Vantage NV -sykemittari.



KUVA 4 Ergoline 800s polkupyöräergometri

KUVA 5 Polar Electro Oy:n Vantage NV-sykemittari

Kun sykkeet on talletettu sykemittarille, ne pitää siirtää FitWare-ohjelmaan. Sykkeiden siirtoon tarvitaan purkulaitetta, joka toimii siltana sykemittarin ja FitWare-ohjelman välillä. Purkulaite kiinnitetään kaapelilla tietokoneen sarjakanvaliityntään ja sykemittari asetetaan purkulaitteen päälle. Sykemittarista siirretään sykkeet langattomasti purkulaitteeseen, joka muuntaa sykkeet johtokoodiksi, jotka lähetetään sarjakanavaa pitkin tietokoneelle. Purkulaitteiden valmistajana toimii myös Polar Electro Oy. Kuvassa 6 on esitetty Polar Electro Oy:n sykepurkulaite.

KUVA 6 Polar Electron purkulaite ja sykemittari

Edellä mainitut järjestelmän komponentit ovat FitWare-järjestelmän laitekomponentteja ja ne ovat myös valmiina saatavilla. Periaatteessa näillä komponenteilla voitaisiin laskea ihmisen maksimaalista hapenottokykyä ja analysoida työn ja harjoituksen rasittavuutta, mutta työ olisi vaivalloista ja virheiden esiintyminen olisi hyvin todennäköistä. Tämän vuoksi järjestelmään liitetään ohjelmistokomponentti, joka analysoi sykemittarilta saaduista sykkeistä ja muista syötetyistä arvoista analyysien tulokset nopeasti ja tarkasti. Ohjelmistokomponentti on ainoa järjestelmän osa, jota ei ole valmiina saatavilla ja tämän vuoksi se oli rakennettava itse. Ohjelma tarvitsee tietenkin myös tietokoneen toimiakseen, joten järjestelmään lisätään myös laitekomponenttina tietokone. Ohjelman kehittelyn aloittaminen oli perusteltua, koska se helpottaa ja tarkentaa tuntuvasti testaustoimintaa ja työn ja harjoituksen rasituksen analysointia. Lisäksi ohjelmaan voidaan sisällyttää lisätoimintoja, jotka visualisoivat ja auttavat ymmärtämään erilaisia tuloksia. Tärkeä perustelu ohjelman kehittelylle oli tietenkin myös se, ettei vastaavanlaisia ohjelmia, eikä siis myöskään järjestelmiä, ole tällä hetkellä olemassa.

FitWare-ohjelma on luonteeltaan asiantuntijajärjestelmän (AJ) kaltainen. Asiantuntijajärjestelmä on Taarin mukaan tietokoneohjelma, joka toimii rajatulla ongelma-alueella ih-

misasiantuntijan tavoin. Se voi ratkaista monimutkaista päättelyä vaativia ongelmia tai antaa neuvoja sekä usein myös perustelee ratkaisujaan. Hyvin sovellettu asiantuntijajärjestelmä nopeuttaa tulokseen pääsyä, parantaa tuloksen laatua, opetaa käyttäjäänsä ja tuo kustannussäästöjä. Taari määrittelee lisäksi, että asiantuntijajärjestelmä koostuu päättelyn sisältävästä päättelyosasta, sovelluksen tiedot sisältävästä tietämyskannasta ja selkeän käytön mahdollistavasta käyttäjäliittymästä. Mielestäni FitWare-ohjelma täyttää kaikki nämä AJ:n tunnuspiirteet. (7, s. 2 - 5.)

3.4 Järjestelmän kohderyhmät

FitWare:n käyttöalue on laaja: terveydenhuollosta kuntoiluun ja kilpaurheiluun. Suomessa pääkohderyhmiä ovat saairaaloiden kuntoutusosastot ja kuntoutuslaitokset, eri oppilaitokset, työterveyshuollon yksiköt, puolustusvoimat ja myös yksityisektorit. FitWarella suoritettava epäsuora testi ei sovellu ilman laajempia tutkimuksia verenkiertoelimistön sairauksista kärsiville. Sykintätasoon vaikuttava lääkitys sotkee testitulosta, kuten muissakin epäsuorissa hapenkulutuksen mittauksissa.

4 FITWARE-OHJELMAN VAATIMUSMÄÄRITTELY

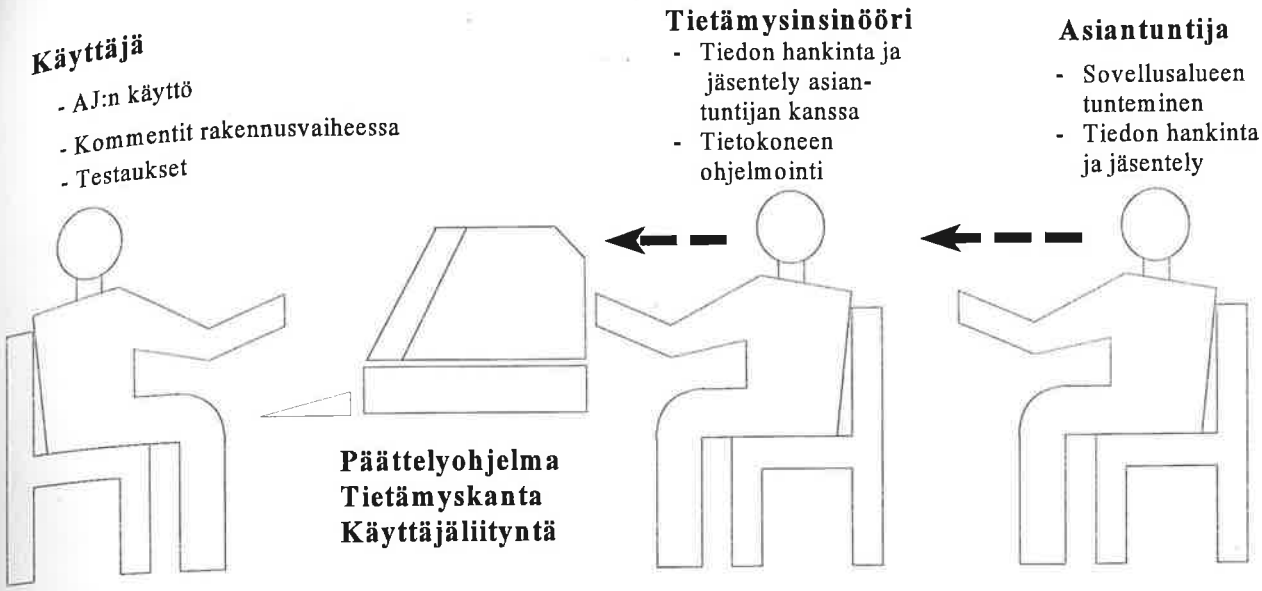
Edellisessä kappaleessa kuvattiin *FitWare-järjestelmän* toimintaa, jonka yhtenä osana toimii *FitWare-ohjelma*, jota voitaisiin kutsua myös FitWare-asiantuntijajärjestelmäksi. Järjestelmän määrittelystä ei kuitenkaan saada vielä tarpeeksi tietoa, jotta voitaisiin aloittaa ohjelman suunnittelu. Tämän vuoksi tarvitaan ohjelman vaatimusmäärittelyä, joka kuvaa tarpeeksi yksityiskohtaisesti mitä ohjelma valmistuttuaan tekee. Lisäksi on tiedettävä muita ohjelmaan liittyviä tekijöitä, kuten ohjelman käyttäjäryhmä ja suorituskyky. Näiden tietojen perusteella voidaan edetä ohjelman suunnitteluun ja toteutukseen. Vaatimusmäärittelyllä ei kuitenkaan määritellä sitä, *miten* ohjelma toteutetaan tai *kuinka* sen suorituskyky saavutetaan.

Tyypillisen asiantuntijärjestelmän kehittelyyn Tuurin mukaan osallistuu tyypillisesti kolme henkilöä (kuva 7):

-asiantuntija, jonka tietoa (tietämystä) siirretään tietokoneella

-tietämysinsinööri, joka suorittaa asiantuntijan tiedon siirtämiseen tietokoneelle

-käyttäjä, joka esittää omia tarpeitaan tulevaa järjestelmää kohtaan ja testaa myös keskeneräistä järjestelmää (7, s. 5).



KUVA 7 Asiantuntijajärjestelmän rakentaminen (6, s. 18)

Käytännössä AJ:n rakentamiseen osallistuvia henkilöitä voi olla enemmänkin kuin kolme. Toisaalta sen joskus tekee yksi tai kaksi henkilöä. FitWare-ohjelman kehityksessä työnantaja toimi sekä asiantuntijana että käyttäjänä. Lisäksi käytettiin ulkopuolisia asiantuntijoita.

FitWare-ohjelman vaatimusmäärittelyn laatimisessa työnantajalla oli paljon ideoita ja "sumeaa materiaalia", joista suodatin oleelliset tiedot erilleen. Työ oli vaikeaa ja vaativaa viestintätyötä yhdessä työnantajan kanssa. Työtä hankaloitti myös se, että en työn alussa tarkkaan tienyt miten pitkään halutun toiminnon toteuttaminen Windows-ohjelmoinnilla veisi aikaa, tai oliko se ylipäätään mahdollista toteuttaa. Suuri osa vaatimusmäärittelytyöstä meni myös ohjelmassa käsiteltävän aihealueen tutustumiseen - asioiden sisäistämiseen sekä niiden siirtämiseen matemaattiseen muotoon. Vaatimusmäärittelyn teossa ei pyritty täydellisyyteen, vaan luomaan perustiedot ohjelmistosuunnitteluun. Koska ohjelmasta ei ollut olemassa aikaisia versioita tai prototyyppejä, oli odotettavissa, että ohjelman kehitysmalliksi tuli ns. evoluutiomalli. Vaatimusmäärittelyn puitteita ja epäselvyyksiä paikkailtiin katselmuksilla sekä sovelluksen mallintamisella ennen varsinaista toteutusvaihetta.

Seuraavissa kappaleissa (4.1 - 4.3) on esitetty ohjelmiston vaatimukset kehitystyön aloitusvaiheessa. Lopulliseen ohjelmaan on lisätty toteutusvaiheen aikana ominaisuuksia, joita ei alkuperäisissä vaatimuksissa ole esitelty. Tämä johtui ohjelman luonteesta (asiantuntijajärjestelmä), sekä siitä, että työnantajalla, joka toimi siis käyttäjänä ja asiantuntijana kehitystyön aikana, ei ollut aikaisempaa kokemusta tietokoneohjelmien kehittämisestä.

4.1 Käyttäjät

Käyttäjäryhmän tunnistaminen on erinomaisen tärkeää suunniteltaessa ohjelmistoja, jotta tiedetään minkälainen on heidän käsityksensä hyvästä ohjelmasta. FitWare-ohjelman käyttäjäryhmiä ovat mm. työterveyslaitokset, kuntoutuslaitokset, sairaalat, terveyskeskukset, urheiluseurat ja yksityisyrittäjät. Epäsuorissa testauksissa ei ole aikaisemmin käytetty tietokoneohjelmia, joten käyttäjien kokemus tietotekniikasta ja tietokoneohjelmista on vähäistä. Tämä seikka on otettava huomioon FitWaren suunnittelussa, etenkin käyttöliittymän osalta. Ohjelma ei saa olla ylivoimaisen vaikea käyttää, mutta ei saa kuitenkaan turhauttaa kokenutta käyttäjää. Tämän vuoksi ohjelmasta on mahdollisesti tehtävä sovitteluratkaisu näiden kahden tavoitteen välillä.

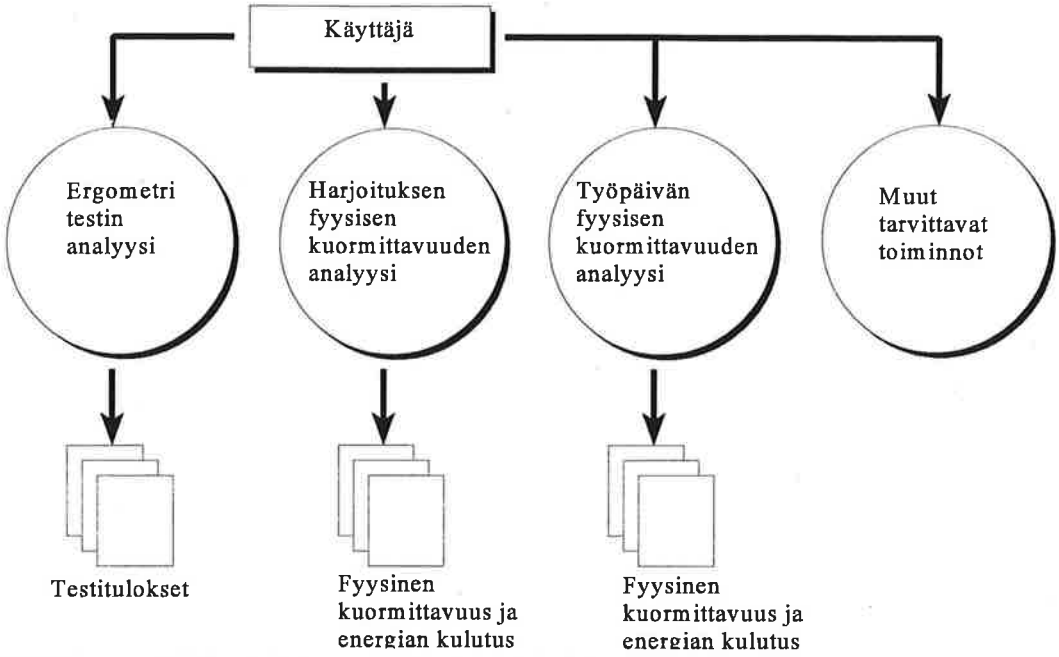
4.2 Toiminnalliset vaatimukset

Toiminnollisilla vaatimuksilla kuvataan ohjelman toiminta. Toiminnalliset vaatimukset on jaettava hierarkisesti, jotta niiden tulkitseminen olisi vaivattomampaa. Ylemmällä tasolla ovat päävaatimukset, joista siirrytään asteittain alemmille tasoille. Tällaista mallinnustapaa kutsutaan "top-down"-suunnitteluksi.

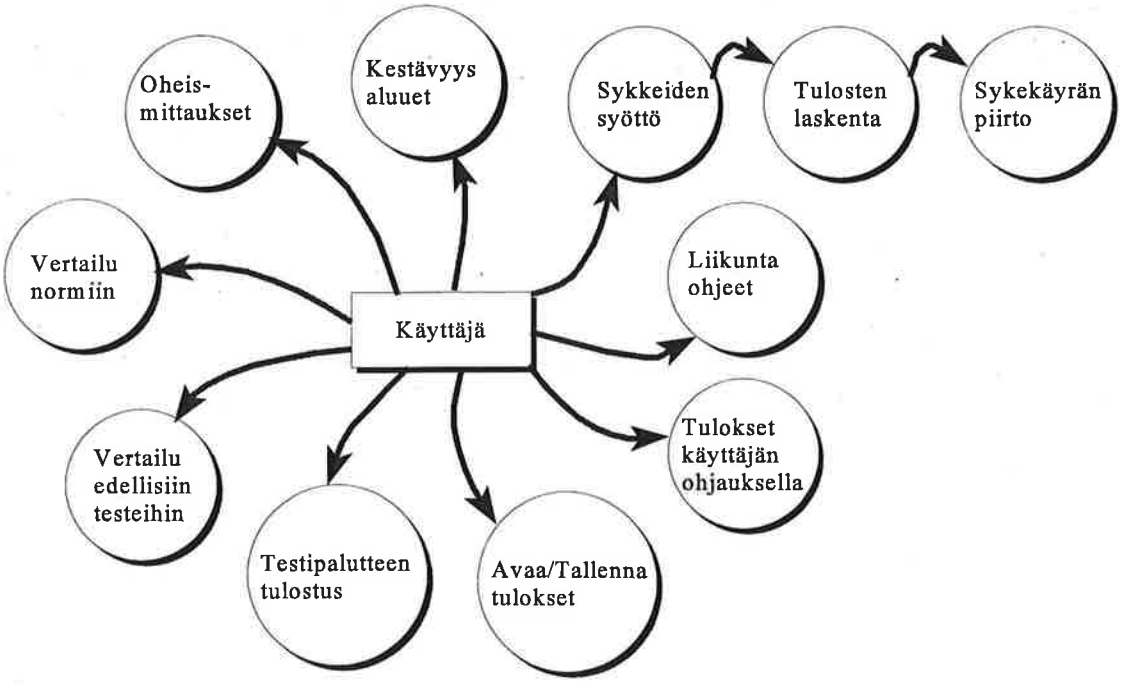
FitWare-ohjelman yksinkertaisin toimintamalli on esitetty kuvassa 8. Käyttäjä voi suorittaa FitWare-ohjelmalla kolme analyysiä: polkupyöräergometritestin analyysin sekä työpäivän ja harjoituksen fyysisen kuormittavuuden analyysit. Ergometritestin analyysistä saadaan tulokseksi aerobinen suorituskyky (maksimaalinen hapenottokyky), harjoituksen fyysisen kuormittavuuden analyysistä saadaan tulokseksi harjoituksen fyysinen kuormittavuus sekä energian kulutus ja työpäivän fyysisen kuormittavuuden analyysistä vastaavasti työpäivän fyysinen kuormittavuus sekä energian kulutus.

Ergometritestin analyysi hienontuu kuvan 9 mukaisesti. Analyysin on syötettävä henkilötietojen lisäksi syketieto testin ajalta. Ohjelma laskee henkilö- ja syketiedoista testin tulokset ja piirtää sykekäyrän sekä tulokset näytölle. Ohjelman käyttäjällä on käytössä erilaisia lisätoiminteita, jotka auttavat testattavan analysoinnissa: käyttäjä voi tarkastella kestävyyden eri aloja, syöttää oheismittauksia testin ajalta ja verrata tulosta normi-arvoihin. Lisäksi käyttäjällä on ominaisuus, jolla hän voi ohjata tulosten laskentaa. Ohjelmasta nähdään liikuntaohjeita, joita suositellaan testattavalle tuloksen perusteella. Testin tulokset voidaan avata ja

tallentaa massamuistiin ja myös tulostaa tulostimella. Tuloksia voidaan verrata vanhoihin testituloksiin.



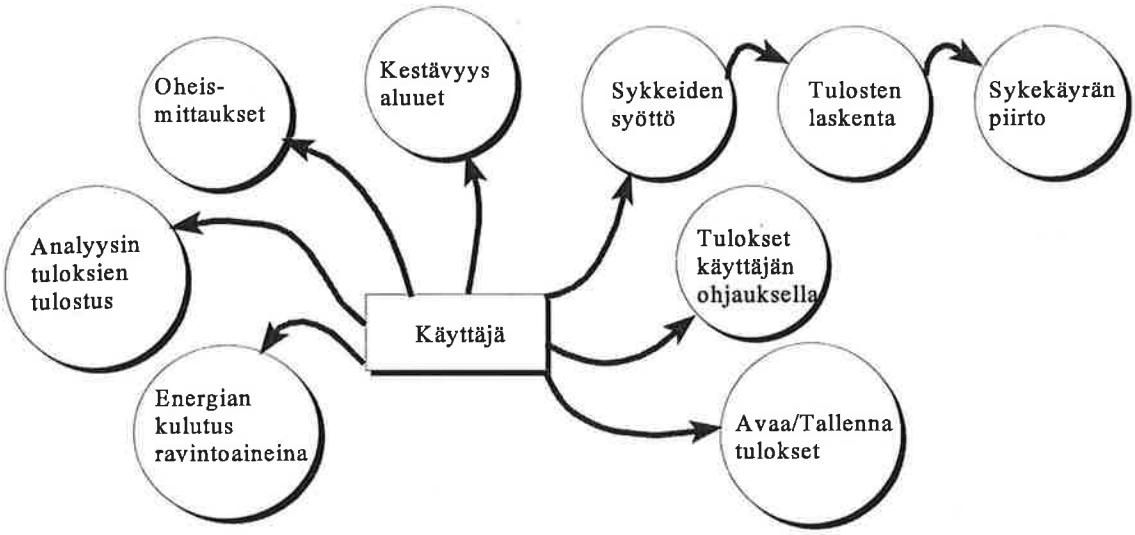
KUVA 8 FitWare-ohjelman yksinkertaisin toimintamalli



KUVA 9 Polkupyöräergometritestin toiminnat FitWare-ohjelmassa

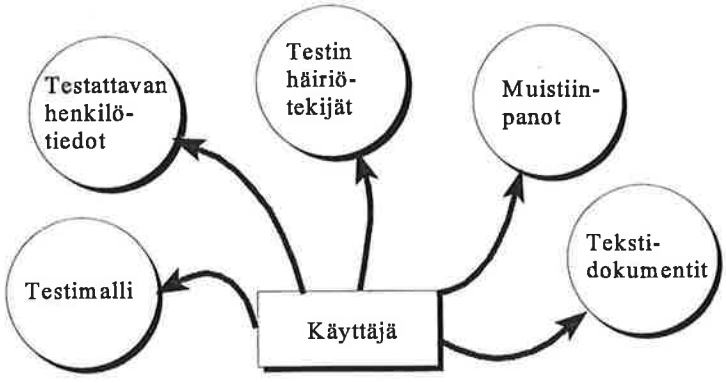
Työpäivän ja harjoituksen fyysisen kuormittavuuden analyysit ovat toiminnaltaan samanlaiset. Kuvassa 10 on esitetty analyysien toimintaa. Analyyseihin syötetään sykkeet ja fyysinen rasittavuus lasketaan suhteessa henkilön maksimaaliseen hapenottokykyyn. Analyyseissä

lasketaan myös energian kulutus, jota voi tarkastella myös ravintoaineiden kulutuksena. Käyttäjä voi ohjata tulosten laskua myös työpäivän ja harjoituksen analyyseissä valitsemalla halutun alueen sykekäyrältä. Analyyseissä on käytössä yksi oheismittaus. Tulokset voidaan avata ja tallentaa massamuistiin sekä myös tulostaa tulostimelle.



KUVA 10 Työpäivän ja harjoituksen fyysisen kuormittavuuden analyysin toiminta FitWare-ohjelmassa

Muut toiminnot, jotka FitWare-ohjelmalta vaaditaan on esitetty kuvassa 11. Välttämättömiä esitietoja polkupyöräergometritestin analyysiä varten ovat testimalli ja testattavan henkilötiedot. Testin häiriötekijöistä testaaja näkee tekijät, jotka heikentävät ergometritestin tulosta. Tekstidokumentit sisältävät tietoa testauksesta ja ihmisen fysiologiaan liittyvistä ilmiöistä.



KUVA 11 FitWare-ohjelman muut tarvittavat toiminnot

4.2.1 Ergometritestianalyysin toimintojen ominaisuudet

4.2.1.1 Sykkeiden syöttäminen

Sykkeet on mahdollista syöttää ohjelmaan sykemittarin avulla tai manuaalisesti näppäimistön kautta. Sykemittarilla syötettäessä käyttäjällä on mahdollisuus valita sykemittarin ja purkulaitteen malli sekä käytettävä sarjakanavaportti. Ohjelman on tuettava sykemittareiden tallennusvälejä 5, 15 ja 60 s. Ergometritestianalyysin on mahdollista purkaa enintään 750 kappaletta yksittäisiä sykkeitä, näin ollen testin maksimikestoksi tulee viiden sekunnin tallennusvälillä 62,5 minuuttia, 15 sekunnin tallennusvälillä 187,5 minuuttia ja 60 sekunnin tallennusvälillä 750 minuuttia.

Manuaalisella sykkeen syötöllä tarkoitetaan, että käyttäjä syöttää sykkeet näppäimistöltä ohjelmaan. Sykkeiden syöttöväli on tuolloin yksi minuutti ja enimmäismäärä sykkeille on 45 kappaletta, eli testin enimmäiskesto voi olla 45 minuuttia.

4.2.1.2 Tulosten laskeminen

Maksimaalinen hapenotto lasketaan seuraavasta yhtälöstä.

$$VO_{2max} = \frac{(P_{max} \cdot 12.48 + 217)}{m} \quad (1)$$

VO_{2max}	maksimaalinen hapenottokyky, ml/kg/min
P_{max}	maksimityöteho, W
m	testattavan paino, kg

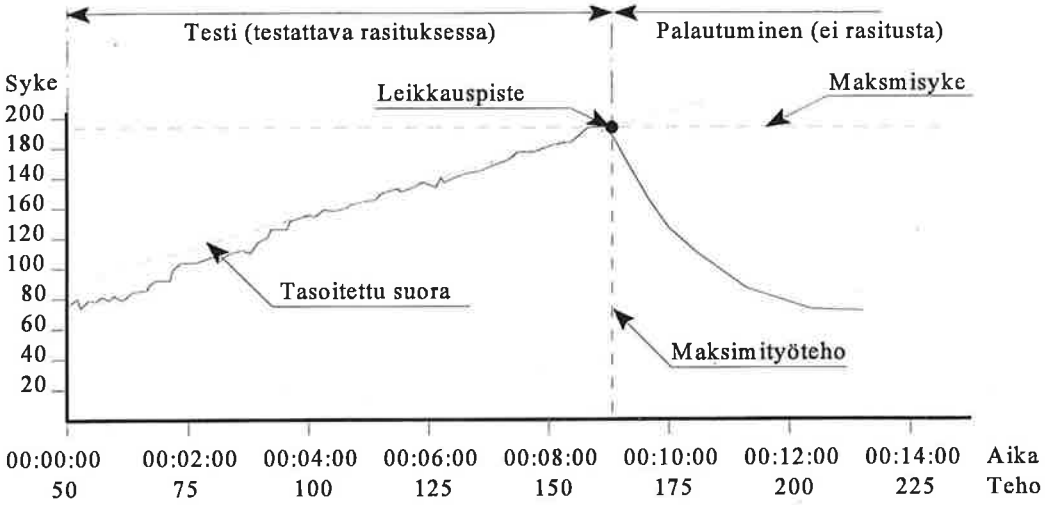
Maksimityöteho saadaan sykekäyrästä, sillä se on henkilön maksimisykettä vastaavan viivan ja sykekäyrän tasoitetun suoran leikkauspisteen x-akselin arvo. Kuvassa 12 on esitetty, kuinka maksimityöteho näkyy sykekäyrässä.

Tasoitettu suora sovitetaan sykekäyrään pienimmän neliösumman menetelmällä. Suora on muotoa y=ax+b. Tekijät a ja b lasketaan pienimmän neliösumman kaavoilla:

$$b = \frac{\sum y_i \sum x_i^2 - \sum x_i \sum x_i y_i}{n\sum x_i^2 - (\sum x_i)^2} \quad (3)$$

$$a = \frac{n\sum x_i\, y_i - \sum x_i \sum y_i}{n\sum x_i^2 - (\sum x_i)^2} \qquad (2)$$

n	sykkeiden kokonaislukumäärä
x_i	aika, s
y_i	syketaajuus, kertaa minuutissa



KUVA 12 Maksimityötehon lukeminen ergometritestin sykekäyrältä

Kun maksmiteho on laskettu, lasketaan maksimaalinen hapenotto kaavalla 1. Tulos esitetään myös yksikkönä litraa minuutissa (l/min), teho painokiloa kohden (W/kg) sekä METs-arvona. Maksimaalinen hapenotto, yksikkönä l/min lasketaan seuraavalla kaavalla.

$$VO_{2max} = \frac{\frac{P_{max} - 40}{10} \cdot 124 + 726}{1000} \qquad (4)$$

VO_{2max}	maksimaalinen hapenottokyky, l/min
P_{max}	maksmityöteho, W

Teho painokiloa kohden lasketaan kaavalla:

$$W/kg = \frac{P_{max}}{m} \qquad (5)$$

W/kg	teho painokiloa kohden, W/kg
P_{max}	maksimityöteho, W
m	testattavan paino, kg.

METs-arvo lasketaan seuraavalla kaavalla.

$$METs = 3{,}5 \cdot VO_{2max} \tag{6}$$

VO_{2max} maksimaalinen hapenottokyky, ml/kg/min

Maksimaalisen hapenottokyvyn tulokset näytetään siis yksikköinä

-ml/kg/min (kaava 1)

-W (kaavat 2 ja 3)

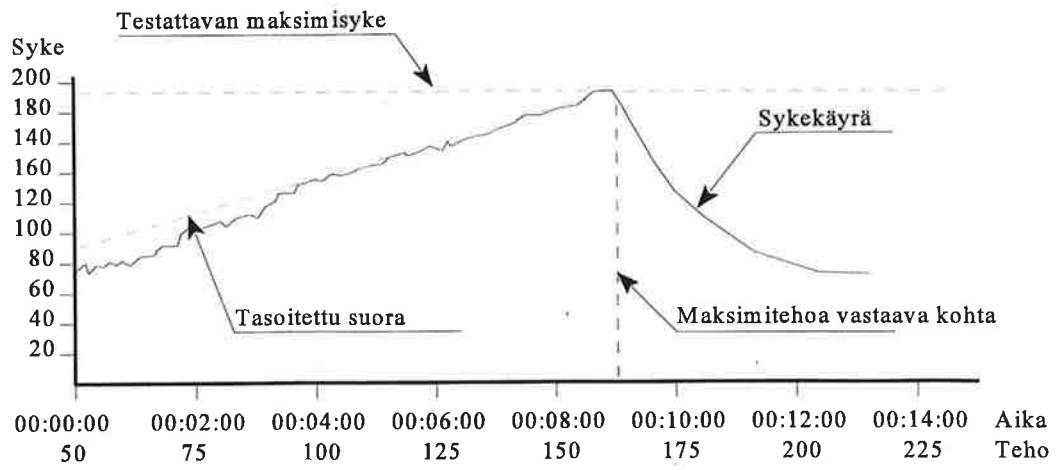
-l/min (kaava 4)

-W/kg (kaava 5)

-METs (kaava 6).

4.2.1.3 Sykekäyrän piirtäminen

Ergometrianalyysissä luetusta syketiedosta on muodostettava graafi, eli sykekäyrä. Mittauksien esittäminen graafisesti on havainnollisempaa kuin pelkästään lukuina ja näin saadaan kahden suureen välinen riippuvuus ja sen laatu näkyviin. Graafisesta esityksestä nähdään nopeasti erikoispisteet (esim. maksimi- ja minimikohdat) ja mahdolliset mittausvirheet. Kuvassa 13 on hahmotelma vaaditusta sykekäyrästä. Käyrä on esitetty kaksi-ulotteisesti siten, että x-akselilla näkyvät aika sekä teho. Aika on muodossa tunnit, minuutit ja sekunnit ja jaotellaan testimallin mukaisesti, esim 00:22:00. Tehot ilmoitetaan watteina ja jaotellaan myös testimallin mukaan. X-akselin nollakohdassa on testin aloituskohta eli aika on nolla, mutta teho on aloitusvastusta vastaava teho. X-akselin pituus määräytyy sykelukumäärän



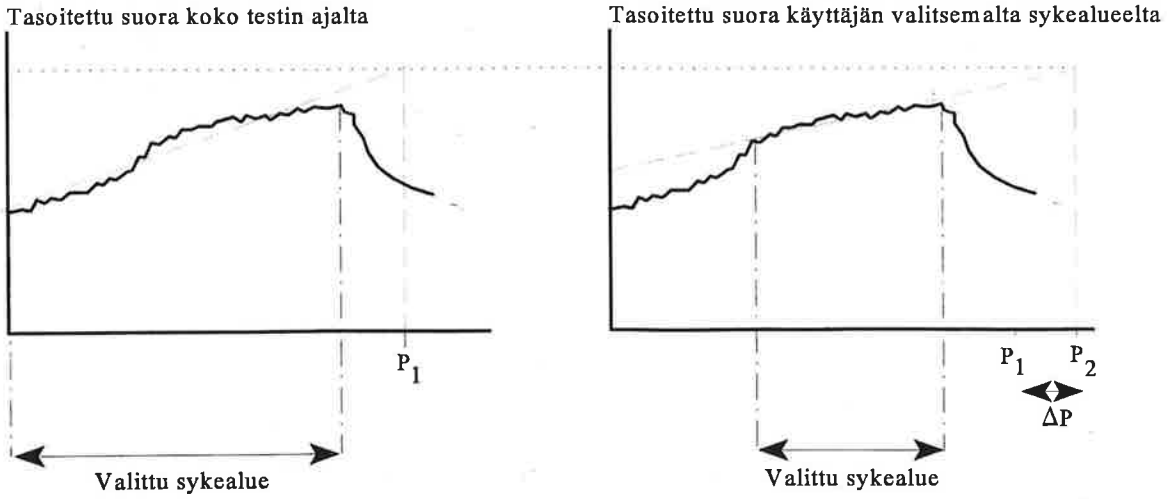
KUVA 13 FitWare-ohjelman ergometrianalyysin sykekäyrä. Aloitusvastus on kuvassa 50 wattia ja testimalli 25 wattia / 2 min

mukaan. Sykekäyrän y-akselilla on syke, joka ilmoitetaan 20 sykkeen välein. Y-akseli muodostuu siten, että alin syke on nolla ja ylin syke on henkilön maksimisyke pyöristettynä ylöspäin seuraavaan sykeväliin. Sykkeen yksikkönä on sykäystä minuutissa (bpm, Beats per minute).

Sykekäyrä-graafissa näkyvät katkoviivoina testattavan maksimisyke, tasoitettu suora ja edellä mainittujen leikkauskohta eli maksimitehoa vastaava kohta. Käyttäjällä on mahdollisuus muuttaa graafin asetuksia. Muutettavissa ovat graafin eri komponenttien värit sekä apuviivaston näkyminen. Jos apuviivasto on näkyvissä, se on jaoteltu y-akselin suhteen samoin kuten sykkeet ja x-akselin suhteen samoin kuten aika.

4.2.1.4 Tulokset käyttäjän ohjauksella

Käyttäjällä on oltava mahdollisuus vaikuttaa itse tuloksien laskentaan. Ohjelmassa on käyttäjän pystyttävä valitsemaan haluttu sykealue sykekäyrältä, johon tasoitettu suora lasketaan. Tämä toimenpide on välttämätön sillä sykekäyrien muoto saattaa vaihdella hyvinkin paljon ja automaattinen käyrän tasoitus ei saata aina toimia oikein. Kuvassa 14 on esimerkkitapaus. Sykekäyrä nousee alussa nopeammin, mutta testin loppua kohden hidastuu. Koko testiajalta laskettu tasoitettu suora ei anna tässä tapauksessa oikeata tulosta. Käyttäjän on valittava itse alue, josta hän haluaa suoran laskettavan. Kuvassa tulosten poikkeamaa on merkitty ΔP:llä. Käyttäjä voi vaikuttaa myös tuloksiin muuttamalla testattavan painoa tai maksimisykettä.



KUVA 14 Sykealueen valinnan merkitys FitWare-ohjelmassa

4.2.1.5 Kestävyysalueet

Ergometritestin analyysistä on saatava graafinen esitys, jossa kuvataan testattavan kestävyysalueet. Kestävyysalueita ovat peruskestävyys-, vauhtikestävyys- ja maksimikestävyysalueet. Esityksestä on nähtävä myös alueita vastaava sykintätaajuus, prosentuaalinen osuus kaikista kestävyyden alueista ja aika, kuinka kauan kyseisellä alueella on liikuttu.

Peruskestävyysalueella tarkoitetaan ihmisen peruskestävyysalueen alarajan ja aerobisen kynnyksen välistä sykealuetta. Vauhtikestävyydellä tarkoitetaan aerobisen ja anaerobisen kynnyksen välistä aluetta ja maksimikestävyydellä anaerobisen ja maksimisykintätaajuuden välistä aluetta. Kestävyysalueet lasketaan polkupyöräergometritestin ajalta - palautumista ei lasketa mukaan, vaikka se näkyisi sykegraafissa.

4.2.1.6 Liikuntaohjeet

Ergometritestien tuloksien perusteella käyttäjä antaa testattavalle liikuntaohjeet. FitWare-ohjelmassa on näytöllä esitettävä liikuntaohjeet, joista selviää miten olisi liikkuttava, jotta voisi parantaa perus-, vauhti- tai maksimikestävyysalueita. Ohjeissa on annettu kestävyysalueita vastaava harjoittelusykealue, harjoittelun tärkeys, harjoituksen kesto ja useus sekä liikuntalajeja, jotka vastaavat harjoittelua. Kestävyysaluetta vastaavat harjoittelusykealueet ovat vastaavat kuin kohdassa 4.2.1.4 esitetyt kestävyysalueiden sykealueet.

Koska testattavien tulokset poikkeavat, on liikuntaohjeiden otsikkoa pystyttävä muuttamaan. Käytössä on kolme eri vaihtoehtoa: liikuntaohje, harjoitteluohje ja liikuntalääke. Otsikon merkitys on puhtaasti psykologinen, mutta se kertoo testattavalle harjoittelun tärkeyden. Huonokuntoisille otsikko on liikuntalääke, kuntoilijoille ja urheilijoille, otsikko on liikuntaohje tai harjoitteluohje.

Käyttäjällä on oltava myös mahdollisuus muuttaa jokaista ohjeen arvoa tai tekstiä. Ohjeissa näkyvät oletusarvot ovat liitteessä 1.

4.2.1.7 Tulosten vertaaminen normiin

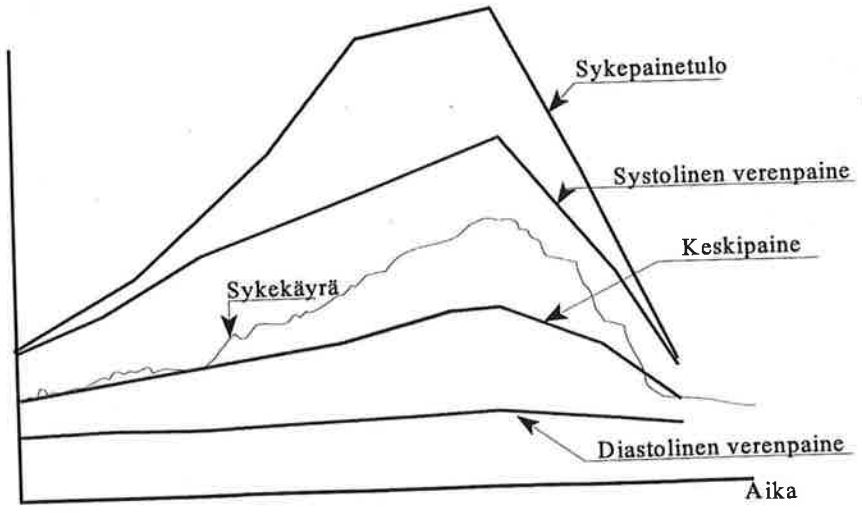
Pelkkä numeerinen arvo ei kerro tarpeeksi hapenottokyvystä, tämän vuoksi ohjelmassa on

oltava normit, johon tulosta verrataan. Normeissa on määritelty numeerista arvoa ja testattavan ikää ja sukupuolta vastaava sanallinen tulos. Normeja varten on oltava erillinen editointitila, johon käyttäjä syöttää lukuarvot, ikävälit ja sanalliset määrittelyt. Lukuarvot voidaan syöttää yksiköissä ml/kg/min, METs, l/min tai W/kg. Normit voidaan avata ja tallentaa massamuistiin.

FitWare-ohjelmassa on oltava graafinen esitys, josta nähdään testattavan kuntoarvio oman ikäluokkansa ja sukupuolensa normistossa. Tämä auttaa testattavaa näkemään oman kuntonsa verrattuna omaan ikäryhmäänsä ja sukupuoleensa. Esityksestä on nähtävä myös keskimääräinen suorituskyky, joka on normin keskiarvo.

4.2.1.8 Oheismittaukset

Ergometritestin analyysissä on oltava mahdollisuus syöttää oheismittauksia testin ajalta. Käytettävät oheismittaukset ovat verenpaine-, BORG- ja PEF-mittaukset. Verenpainemittauksessa on oltava mahdollisuus syöttää 10 kappaletta systolisia ja diastolisia verenpainearvoja testin ajalta. Käyttäjä merkitsee editointitilaan ajan ja sitä vastaavan diastolisen ja systolisen paineen. Verenpaineesta on saatava graafinen esitys, jossa näkyvät sykekäyrä, diastolinen ja systolinen verenpaine, keskipaine sekä sykepainetulo (kuva 15).



KUVA 15 Verenpaineen graafinen esitys FitWare-ohjelman oheismittauksissa

Keskipaine lasketaan systolisesta ja diastolisesta verenpaineesta seuraavasti.

$$MBP = \frac{SBP - DBP}{3} + DBP \quad (7)$$

MBP	keskipaine, mmHg
SBP	systolinen verenpaine, mmHg
DBP	diastolinen verenpaine, mmHg

Sykepainetulo lasketaan seuraavalla kaavalla.

$$DP = HR \cdot SBP \quad (8)$$

DP	sykepainetulo, mmHg/s
HR	syketaajuus, lyöntiä/s
SBP	systolinen verenpaine, mmHg

Lisäksi verenpaineen mittauksissa on esitettävä keskipaine, sykepainetulo, sydänlihaksen hapenkulutus ja minuuttitilavuus testin lopetushetkellä. Sydänlihaksen hapenkulutus lasketaan kaavalla

$$MO_2 = \frac{(0{,}14 \cdot HR \cdot SBP) - 6{,}3}{100} \quad , missä \quad (9)$$

MO_2	sydänlihaksen hapenkulutus, ml/100 g sydänlihasta
HR	syketaajuus, lyöntiä/min
SBP	systolinen verenpaine, mmHg.

Minuuttitilavuuden laskemisessa käytetään kolmea eri kaavaa: miehelle ja naiselle eri kaava sekä sydänsairautta potevalla miehelle oma kaava. Minuuttitilavuus terveelle miehelle lasketaan seuraavalla kaavalla.

$$CO = 5{,}3 + 4{,}6 \cdot VO_{2max} \quad (10)$$

CO	minuuttitilavuus (Cardiac Output), l/min
MAX.VO_2	maksimaalinen hapenottokyky, l/min

Sydänsairaalla miehelle käytetään kaavaa

$$CO = 3{,}1 + 5{,}9 \cdot MAX.VO_2. \quad (11)$$

Naiselle minuuttitilavuus lasketaan kaavalla

$$CO = 4{,}7 + 4{,}0 \cdot MAX.VO_2. \quad (12)$$

BORG-mittauksella mitataan testattavan *koettua* kuormittavuutta ergometritestin aikana sekä lopetushetkellä. Testattava kertoo kokeneensa kuormituksen asteikolla 6-20, missä 6 vastaa kevyttä kuormitusta ja 20 raskasta kuormitusta. BORG-indeksiä vastaavat sanalliset määrittelyt ovat esitetty liitteessä 2. BORG-arvoista on saatava graafinen esitys, josta näkyy testin kuorma watteina ja testattavan koettu kuormitus BORG-indekseinä. Ohjelmassa on varattu 45 mittauspistettä BORG-arvolle.

PEF (Peak Flow)-arvolla mitataan uloshengityksen huippuvirtausta. Mittauspisteitä ovat mittaus levossa ja heti rasituksen lopusta viiden minuutin välein 30 minuuttiin asti. Mittaustuloksista on laskettava suhde lepo-PEF:iin prosentteina. Tuloksista on muodostettava myös graafinen esitys, jossa y-akselilla kuvataan PEF-arvot ja x-akselilla vastaava aika.

4.2.1.9 Tulosten vertaaminen edellisiin testeihin

Ergometritestin analyysin tuloksia on pystyttävä vertailemaan testattavan edellisiin mittaustuloksiin. Vertailulla nähdään, mihin suuntaan henkilön hapenottokyky on muuttunut. Vertailussa on pystyttävä vertailemaan keskenään mahdollisemman montaa edellistä testiä nykyiseen testiin. Vertailtavia tuloksia ovat ergometritestin tulokset (kuntoarvio, METs, ml/kg/min, l/min, W/kg ja W), kestävyysalueet, palautuminen, painoindeksi (BMI), verenpaine (systolinen, diastolinen, keskipaine ja sykepainetulo) ja PEF-tulokset. Vertailussa on nykyiset ja edelliset tulokset näytettävä absoluuttisina arvoina sekä suhteellisina muutoksina. Muutos edelliseen testiin näytetään prosentteina. Tulokset on myös esitettävä graafisesti, jotta niiden tulkitseminen olisi havainnollisempaa.

4.2.2 Työpäivän ja harjoituksen fyysisen kuormittavuuden analyysien toiminto jen ominaisuudet

4.2.2.1 Sykkeiden syöttäminen

Työpäivän ja fyysisen kuormittavuuden analyyseissä sykkeiden syöttäminen FitWare-ohjelmaan tapahtuu ainoastaan sykemittareiden avulla. Syketallennusvälit voivat olla 5,15 ja 60 s

ja sykkeitä voidaan analyysin purkaa 8000 kpl. 5 sekunnin tallennusvälillä harjoituksen tai työpäivän maksimikestoksi tulee 11,1 tuntia, 15 sekunnin tallennusvälillä maksimikesto on 33,3 tuntia ja 60 sekunnin tallennuksella 133,3 tuntia. Muulta osin sykkeiden syöttäminen on määritelty samoin kuin ergometritestin analyysissäkin.

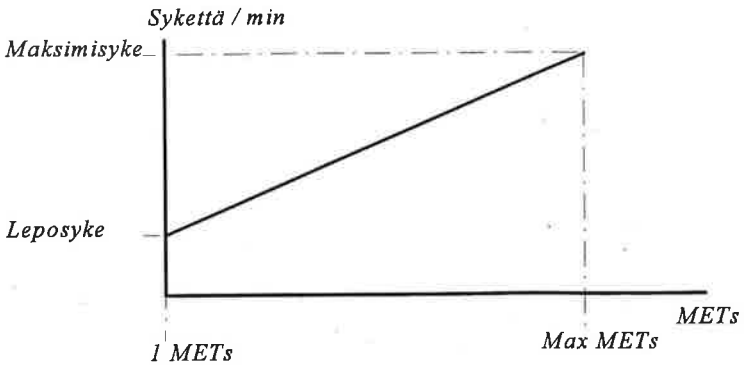
4.2.2.2 Tulosten laskeminen

Työpäivän ja harjoituksen analyyseistä lasketaan fyysinen kuormittavuus sekä energian kulutus. Fyysinen kuormittavuus lasketaan METs-yksikkönä sekä suhteellisena osuutena maksimaalisesta hapenottokyvystä. Tulokset lasketaan sykintätaajuuden keskiarvosta. Laskemisessa käytetään hyväksi tietoa, että 1 METs vastaa ihmisen leposykettä ja maksimi-METs-arvo vastaa maksimisykettä (maksimi METs-arvo saadaan ergometritestin analyysistä). Tämän tiedon perusteella voidaan suoran yhtälön avulla laskea, mitä METs-arvoa keskisyke vastaa ja sen jälkeen, kuinka paljon se on verrattuna maksimaaliseen hapenottokykyyn. Kuvassa 16 on kuvattu sykkeen ja METs-arvon välistä riippuvuutta. Koska METs-arvo ja syketaajuus ovat siis lineaarisesti toisiinsa verrannollisia, voidaan METs-arvo laskea syketaajuuden funktiona seuraavasti:

$$METs = f(HR) = HR \cdot k + b \quad , \; missä \tag{13}$$

$$b = k \cdot HR_{lepo} - 1. \tag{14}$$

$$k = \frac{METs_{max} - 1}{HR_{max} - HR_{lepo}} \tag{15}$$



KUVA 16 METs-arvo kasvaa lineaarisesti ihmisen syketaajuuden mukaan

Kulutettu energia lasketaan seuraavasta yhtälöstä.

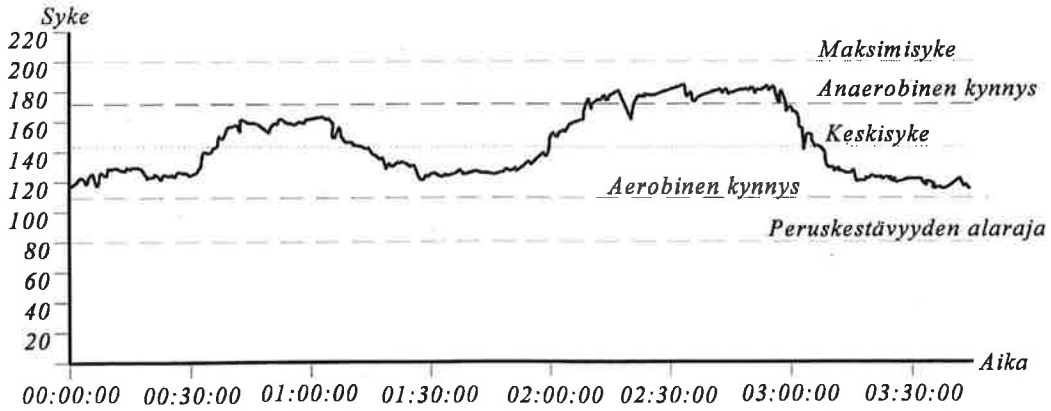
$$E = \frac{VO_2 \cdot m}{T} \cdot T \qquad (16)$$

E	energia, kcal,
VO_2	hapenottokyky, METs
m	testattavan paino, kg
T	aika, h

Energia ilomoitetaan ohjelmassa sekä kilokaloreina (kcal) että kilojouleina (kJ).

4.2.2.3 Sykekäyrän piirtäminen

Työpäivän ja harjoituksen fyysisen kuormittavuuden analyyseissä sykekäyrä esitetään kuvan 17 mukaisesti. Y-akselilla ovat sykkeet ja x-akselilla aika. Sykekäyrästä on nähtävä myös henkilön kynnysalueet sekä sykealueen keskiarvo. Käyttäjällä on oltava myös mahdollisuus lähentää (zoomata) sykekäyrää, jotta sykekäyrän tarkastelu olisi havainnollisempaa. Muut sykekäyrän ominaisuudet määrittellään samoin kuin ergometritestianalyysin sykekäyrälläkin (ks. 4.2.1.2).

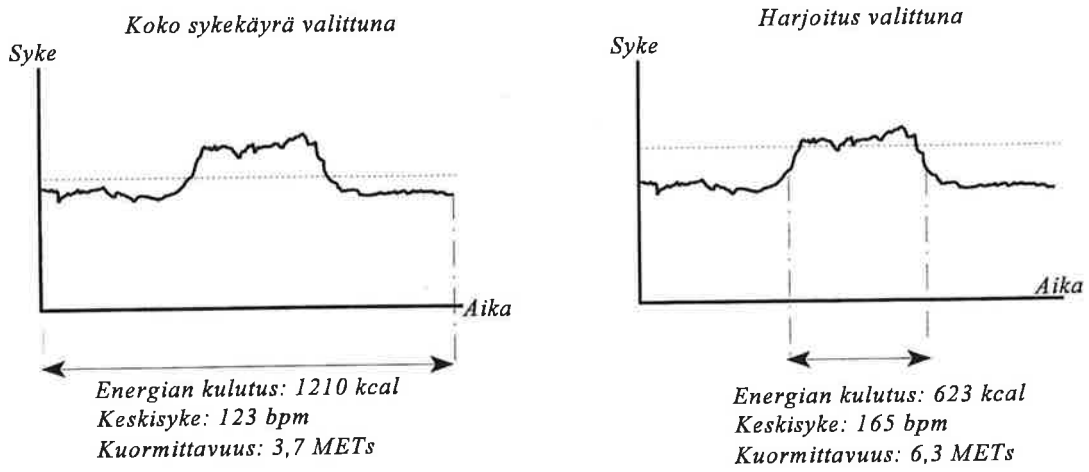


KUVA 17 FitWare-ohjelman työpäivän ja harjoituksen fyysisen kuormittavuuden analyysien sykekäyrä

4.2.2.4 Tulokset käyttäjän ohjauksella

Käyttäjä voi valita työpäivän ja harjoituksen analyyseissä sykekäyrältä alueen, jolta tulokset lasketaan. Jos esimerkiksi sykemittarille on tallennettu kolme tuntia sykkeitä, joista tunti on

käytetty harjoittelussa, käyttäjä voi sykekäyrältä valita tämän harjoitusalueen erikseen ja tarkastella sen kuormittavuutta ja energian kulutusta (kuva 18).



KUVA 18 Sykekäyrältä voidaan valita sykealue, josta analyysin tulokset lasketaan. Kuvassa olevat lukuarvot ovat vain suuntaa antavia

4.2.2.5 Kestävyysalueet

Kestävyysalueet ilmoitetaan samoin kuten ergometritestianalyysissäkin (ks. 4.2.1.5). Kestävyysalueet lasketaan kuitenkin käyttäjän määrittelemältä alueelta (ks. 4.2.2.4). Jos käyttäjä ei ole määritellyt aluetta, laskentaan käytetään koko sykekäyrän aluetta.

4.2.2.6 Energian kulutus ravintoaineina

Jotta energian kulutusta voitaisiin hieman konkretisoida, ilmoitetaan energian kulutus myös elintarvikkeiden kulutuksena. Näin voidaan esittää esimerkiksi kuinka monta hampurilaista energian kulutus vastaa. Energian kulutus lasketaan käyttäjän määrittelemältä alueelta. Jos aluetta ei ole määritelty, käytetään koko sykekäyrän aluetta energian laskemiseksi. Analyyseissä vaaditut elintarvikkeet ja niitä vastaavat enegiasisällöt ovat esitetty liitteessä 3.

4.2.2.7 Oheismittaukset

Työpäivän ja harjoituksen analyyseissä käytettään vain yhtä oheismittausta: koetun kuormittavuuden mittausta. Koettu kuormittavuus ilmaistaan samalla mitta-asteikolla kuin Borg-mittauksetkin (ks. 4.2.1.7 ja liite 2). Koetulla kuormittavuudella testattava ilmaisee, kuinka

kuormittavalta työpäivä tai harjoitus on tuntunut - mittaus ei anna mitään absoluuttista arvoa. Jos esimerkiksi töissä on mennyt huonosti työilmapiirin takia (mutta työ itsessään ei ole ollut kuormittavaa), henkilö tuntee työpäivän kuormittavana.

4.2.3 Muut tarvittavat toiminnot

Muita FitWare-ohjelmassa tarvittavia toimintoja ovat
-henkilötietojen syöttö
-testimallien syöttö
-muistiinpanojen syöttö
-testin häiriötekijöiden syöttö
-tekstidokumenttien esitykset
-tulostus
-tallennus ja avaus.

Henkilötiedot ovat välttämättömiä, jotta henkilö voitaisiin analysoida. FitWare-ohjelmaan on pystyttävä syöttämään henkilön perustiedot (nimi, ikä, sos.turvatunnus jne.), fyysiset ominaisuudet (pituus, paino, maksimisyke jne.) ja elintavat (tupakointi, liikunta jne.). Näiden lisäksi ihmisen pituuden ja painon suhdetta eli BMI (Body Mass Index)-arvoa on havainnollistettava graafisella esityksellä. Ohjelmassa vaaditut henkilötiedot on esitetty liitteessä 4.

Testimallilla määritellään polkupyöräergometritestissä käytettävän ajan ja kuormituksen suhde sekä aloitusvastus. Testimalli ilmoitetaan yleensä muodossa: aika (min) x teho (W). Esimerkiksi merkintä "2 min x 25 W" tarkoittaa, että kuormitusta lisätään 25 wattia kahden minuutin välein. Aloitusvastuksella ilmoitetaan, millä kuormalla testi aloitetaan. Yleensä aloitusvastuksena käytetään 50 wattia. FitWare-ohjelmassa käytettävä aikaporras voidaan valita väliltä 1...30 minuuttia ja tehoporras väliltä 5...200 wattia. Aloitusvastus voidaan valita väliltä 10...200 wattia.

Ohjelman käyttäjällä on oltava FitWare-ohjelmassa tila, johon hän voi kirjoittaa kommentteja ja huomautuksia analyyseistä. Muistiinpanot on pystyttävä tallentamaan yhdessä muiden tuloksien kanssa saman tiedosto alle.

FitWare-ohjelmassa on pystyttävä merkitsemään ergometritestin häiriötekijöitä, jotka vaikuttavat testitulokseen. Testihuoneen lämpötila ja kosteus saattavat vaikuttaa tulokseen. Jos testattava on ennen testitapahtumaa tupakoinut, juonut kahvia, harrastanut liikuntaa, ottanut lääkkeitä tai ollut flunssassa, ei testitulos myöskään välttämättä ole totuuden mukainen. Häiriötekijät on tallennettava muistiinpanojen tavoin yhdessä muiden tuloksien kanssa. Nämä tekijät käyttäjä voisi kirjoittaa myös muistiinpanoihin, mutta on järkevämpää, että tekijöille on varattu oma paikkansa. Näin käyttäjä paikantaa nopeammin avatusta testistä, vaikuttaako testitulokseen mitään häirötekijöitä.

FitWare-ohjelmaan on liitettävä tekstidokumentteja, joita käyttäjä voi lukea näytöltä. Nämä tekstit eivät anna ohjeita ohjelman käytöstä vaan tietoja testaukseen ja ihmisen fysiologiaan liittyvistä ilmiöistä. Vaaditut tekstidokumentit ovat selvitys METs-, W- ja ml/kg/min-yksiköistä, selvitys fyysisen harjoittelun aiheuttamista fysiologisista vaikutuksista, sekä asiantuntijamääritykset työn fyysisestä kuormittavuudesta.

FitWare-ohjelmasta on pystyttävä tulostamaan erilaisia tulosteita. Jokaisesta analyysistä on pystyttävä tulostamaan testitulokset. Jokaisesta ohjelman graafisesta esityksestä on myös saatava tuloste. Lisäksi on saatava tulosteet liikuntaohjeista ja kestävyysalueista. Jotta käyttäjän olisi helpompi täyttää testattavan henkilötietoja, ohjelmasta on oltava mahdollisuus tulostaa ennakkokyselykaavake, jonka testattava täyttää ennen testiin tuloa. Näiden lisäksi on vielä pystyttävä tulostamaan harjoittelupäiväkirja testattavalle. Liitteissä 6 - 10 on hahmotelmat tulosteista.

FitWare-ohjelmassa on pystyttävä tallentamaan kaikki testattavaan liittyvät tiedot kertatallennuksella yhtenä tiedostona. Tallentaminen tehdään joko kovalevylle tai levykeasemalle. Tallentamisen on oltava ns. tilannetallennus, eli kun käyttäjä avaa tallentamansa tiedoston, hän on ohjelmassa samassa tilanteessa, kun tallennushetkellä. FitWare-ohjelman tiedostoformaatin ei tarvitse olla yhteensopiva minkään muun tietokoneohjelman kanssa.

4.3 Liittymät

Käyttöliittymästä ei ole tarkkaa vaatimusmäärittelyä, vaan siitä sovitaan yksityiskohtaisemmin katselmuksien yhteydessä. Muutamia perusvaatimuksia voidaan kuitenkin asettaa. FitWare-ohjelman on toimittava Windows-käyttöjärjestelmässä. Käyttöliittymän on oltava

graafinen ja mahdollisimman pitkälle hiirikäyttöinen. Ohjelman kaikki toiminnot eivät saa olla yhdessä ikkunassa, vaan toiminnot on jaettava eri ikkunoihin, joiden välinen siirtyminen on oltava mahdollisimman selkeää. Sivujen samanaikaista toimintaa pitää välttää, jotta käyttäjä ei eksyisi ohjelman käytössä. Ohjelmassa käyttetävät graafiset symbolit, painonapit ja graafiset esitykset on oltava selkeitä ja yksiselitteisiä. Liiallista värien käyttöä pitää välttää, jotta ohjelmasta ei tulisi liian "karkkimaista". Käyttöliitymän suunnittelussa on otettava ensisijaisesti huomioon käyttäjien kokemus tietokoneohjelmien käytöstä.

FitWare-ohjelman *laitteistoliittymiä* ovat näppäimistö, hiiri ja sykepurkulaite. Näppäimistön ja hiiren liittyminen tapahtuu Windows-rajapinnan kautta, joten niiden ohjaukseen ei tarvitse erikseen puuttua. Sen sijaan sykepurkulaitteen ja ohjelmiston välisen liitynnän ohjaus on hoidettava ohjelmallisesti. Sykepurkulaite kytketään sarjakanavan kautta tietokoneeseen. FitWare-ohjelmiston ja sykepurkulaitteen välisestä toiminnasta ei ole erityisiä vaatimusmäärittelyjä, kunhan yhteys vain saadaan toimimaan virheettömästi ja yhteys ei häiritse muita käytettäviä ohjelmia.

5 FITWARE-OHJELMAN ARKKITEHTUURIN SUUNNITTELU

Ennen varsinaista suunnittelua selvitin, kuinka sykemittarit saataisiin sovitettua ohjelmaan. Tämän jälkeen valitsin FitWare-ohjelmalle ohjelmointikielen, jotta tietäisin miten suunnittelua olisi jatkettava. Päädyttyäni sovelluskehittimen käyttöön, päätin aloittaa ohjelman suunnittelun mallintamalla ohjelman toimintoja, eli rakentamalla ohjelmasta prototyyppejä. Niiden avulla saatiin ohjelman lopullinen toiminta ja käyttöliittymä määriteltyä.

5.1 Sykemittareiden sovitus ohjelmaan

Työn aloitushetkellä tarkoituksena oli saada Polar Electro Oy:n SportTester-sykemittari toimimaan FitWare-ohjelman kanssa. Otimme yhteyttä laitevalmistajaan, ja saimme sykkeenpurkuun liittyvät dokumentit käyttöömme. Osoittautui, että sykkeiden siirto SportTesteriltä tietokoneelle tapahtuu yksisuuntaisesti ja lähetettävä tieto on ASCII-koodia. Niinpä tulevalta ohjelmointikieleltä ei vaadittu muuta kuin, että se pystyisi ohjamaan ja lukemaan sarjaliityntäporttia.

Työn valmistumisvaiheessa Polar Electro Oy julkaisi kolme uutta kelloa: Vantage NV,

Accurex Plus ja Xtrainer Plus, sekä uuden sykkeiden purkulaitteen Polar Advantage Interface. Vantage NV ja Advantage Interface käyttävät kaksisuuntaista tiedonsiirtoa, ts. kello lähettää tietokoneelle viestiä ja tietokone lähettää kellolle viestiä. Tämä tarkoitti sitä, että Vantage NV:n sovittaminen FitWare-ohjelmaan vaati sykemittareiden uudelleen ohjelmoimista. Kun Polar vielä ilmoitti lopettavansa vanhojen SportTester-kellojen valmistuksen, ei ollut muuta vaihtoehtoa kuin aloittaa kellojen sovitus uudelleen ohjelmaan.

Tällä kertaa Polar ei antanut yksityiskohtaista tietoa sykkeiden purkamisesta ja siihen liittyvistä protokollista, vaan tarjosi valmista DLL-kirjastoa, jonka apuohjelmia kutsumalla voitiin ohjata kaikkia Polarin valmistamia kelloja ja sykepurkulaitteita. Polar käyttää DLL-kirjastosta nimitystä Polar Software Platform. Polar Software Platform on maksullinen kirjastotiedosto, jonka voi ostaa kuka tahansa, jolla on tarvetta sovittaa Polar-sykemittareita tietokoneohjelmaansa.

Sovitus tämän jälkeen ei kuitenkaan vielä ole kovin helppoa, sillä Platformin mukana tuleva ohjekirja ei kerro mitään muuta kuin funktiokutsut ja niiden parametrit - minkäänlaista mainintaa siitä, missä järjestyksessä funktioita kutsutaan ei ole. Onneksi Platformin mukana tuli C++-kielinen esimerkkiohjelman, joka antoi hieman selvyyttä asiaan, ja lopulta kellot ja purkulaite saatiin sovitettua ohjelmaan. Hyvää onnea oli myös se, että ohjelmointikieleksi jo aikaisemmin valittu Visual Basic tuki ulkopuolisen DLL-kirjaston käyttöä.

5.2 Ohjelmointikielen valinta

Ennen suunnittelun aloitusta valittiin ohjelman tuleva ohjelmointikieli, ja etenkin se, käytetäänkö C/C++-kieltä vai jotain sovelluskehitintä. Ohjelmointityökaluksi valittiin Microsoftin sovelluskehitin Visual Basic 3.0 Pro. Valintaan vaikuttivat ohjelman matala oppimiskynnys ja sovellusten synnytämisnopeus. Lisäksi VB:n 3.0 Pro-versiosta löytyi tarpeeksi laitelähei-siä komponentteja, joten sykemittareiden laiteohjaus alussa oli mahdollista toteuttaa ilman C-kieltä. C/C++:n käyttäminen ei tullut kysymykseen, koska C-ohjelmointi Windows-ympäristöön olisi vienyt liikaa aikaa ja rahaa, ja koska C-kielen tuoma nopeus ei ollut FitWare-ohjelmalle tärkeä kriteeri.

Visual Basicin valintaan vaikutti myös se, että ohjelman mallintaminen kehityksen alkuvaiheessa on hyvin nopeata ja helppoa - ohjelman toimintaa voidaan mallintaa periaatteessa

kirjoittamatta ainuttakaan koodiriviä. Sovelluksen pääpiirteitä jäljittelevä prototyyppi auttaa sekä käyttäjiä että toteuttajia ymmärtämään valmistuvaa sovellutusta paremmin. Lopullisen protyypin valmistuttua, toimintojen ohjelmointi Visual Basicillä on nopeata ja helppoa. Prototyyppien rakentaminen oli tärkeää FitWare-ohjelman kehityksessä, koska vaatimusmäärittelyt olivat puutteelliset ja puuttui tarkka näkemys lopullisesta ohjelmasta.

Visual Basicin valintaa puolsi myös se, että muutosten tekeminen jälkeenpäin on paljon helpompaa kuin esim. C/C++:ssa. Ohjelman ensiversioihin tulee aina paljon muutoksia ja virheitä, jonka vuoksi on järkevämpää käyttää aluksi sovelluskehitintä työkaluna kuin C/C++:aa.

Suurimpana haittapuolena Visual Basicissä on sitoutuminen yhteen käyttöjärjestelmään eli Windowsiin. Sovelluksia ei voi siirtää käyttöjärjestelmien välillä yhtä helposti kuin C/C++:ssa, jonka siirrettävyys on erinomainen. Toinen haittatekijä Visual Basicissä on aidon konekielikäännöksen puuttuminen. Visual Basic perustuu tulkkiin, ja sen tuottama EXE-tiedosto ei ole konekieltä vaan tulkattua p-koodia. Tämän takia VB-sovellutukset ovat konekielikäännettyjä ohjelmia hiukan hitaampia. Nämä tekijät eivät kuitenkaan paljoa vaikuttaneet kehitinohjelman valintaan, sillä FitWare-ohjelma oli alunperin tarkoitettu ainoastaan Windows-ohjelmaksi ja toimintanopeudella ei ollut suurta merkitystä ohjelman kokonaisuuden ja käyttömukavuuden kannalta.

Toisena vaihtoehtona sovelluskehittimeksi olisi ollut Borlandin Delphi, joka olisi tarjonnut samat ominaisuudet kuten Visual Basic ja lisäksi olisi tuottanut nopean konekielikoodisen EXE-tiedoston ja tarjonnut kehittyneempää ja laiteläheisempää ohjelmointikieltä. Delphin poisjääntiin vaikutti se, että FitWare-ohjelman alkuprotoilun aikana Delphi ei ollut vielä kovin tunnettu ohjelma Suomessa ja sen olemassaoloa ei yksinkertaisesti huomattu tarpeeksi ajoissa ohjelmistokehityksen kannalta. Delphiin olisi voitu toki siirtyä kesken ohjelmakehityksen, mutta sen tuomia etuja FitWare-ohjelmalle ei katsottu siirtymisen arvoisiksi.

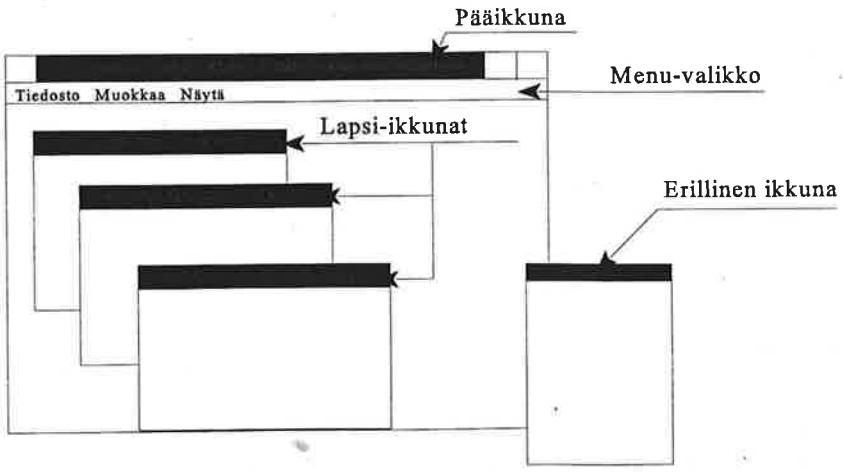
5.3 Ohjelman mallintaminen

Aloitin mallintamisen valitsemalla sovellukselle toimivan käyttöliitymätyypin. Seuraavaksi jaoin vaaditut toiminnot hierarkkisiin osakokonaisuuksiin ja sijoitin ne ikkunoihin. Ikkunoihin sijoitin toimintojen toteuttamiseen tarvittavat komponentit (näkyvät oliot). Tämä kaikki

ei tapahtunut yhdellä yrittämällä vaan ohjelmasta tehtiin useampi prototyyppi ja ne katselmoitiin yhdessä työnantajan kanssa. Tällä tavoin lopullinen ohjelman rakenne ja käyttöliittymä alkoi hahmottua.

5.3.1 Käyttöliittymätyypin valinta

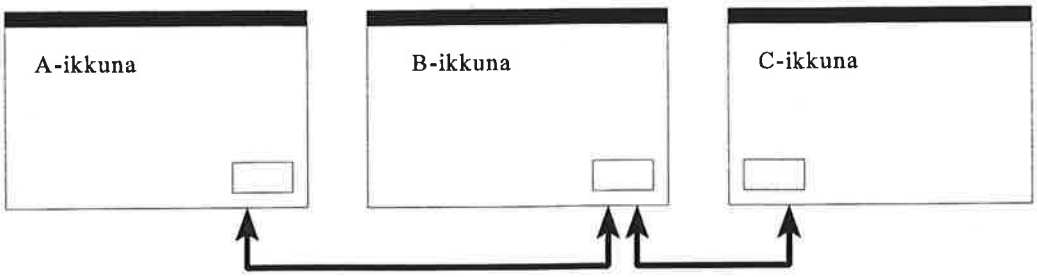
Windows-ympäristössä käyttöliittymä on joko MDI (Multi Document Interface)- tai SDI (Single Document Interface)-tyyppinen. MDI-käyttöliittymässä on yksi pääikkuna, jolla on yksi tai useampi lapsi-ikkuna (Child window) (kuva 19). Käytössä voi olla lisäksi myös erillisiä ikkunoita, joita ei ole sidottu pääikkunaan. MDI-liittymässä on usein käytössä ylävalikko eli ns. menu-valikko, jossa sijaitsevat komennot lapsi-ikkunoiden hallintaan. Samanlaisia asiakirjoja eli lapsi-ikkunoita voi avata useampia pääikkunaan, kuten useissa tekstinkäsittelyohjelmissa. MDI-käyttöliittymän huonona puolena ovat, että sovelluksissa, missä käytetään useita erilaisia ikkunoita, kokematon käyttäjä joutuu usein "kahlaamaan" eri ikkunoiden välillä ja joutuu usein eksyksiin. MDI-liittymää on myös vaikea saada ohjaamaan käyttäjää, sillä MDI perustuu valinnanvapauteen eli siihen, että käyttäjä voi vapaasti valita, mitä ikkunoita on näkyvissä missä tilanteessa tahansa. Myös monet ohjelmistotekniikan asiantuntijat, kuten esim. Bruce Mckinney (8, s. 28), pitävät MDI-käyttöliittymää epämiellyttävänä käyttää. Henkilökohtaisesti pidän MDI-käyttöliittymää erittäin sopivana ratkaisuna useisiin ohjelmiin, mutta myönnän kyllä, että kaikissa sovelluksissa se ei ole paras käyttöliittymäratkaisu, varsinkaan aloittelijoille.



KUVA 19 MDI-tyyppinen käyttöliittymä Windows-ympäristössä

SDI-käyttöliittymätyypissä ei ole hierarkkista ikkunointitapaa, kuten MDI:ssä, vaan kaikki

käytettävät ikkunat ovat erillisiä, toisistaan riippumattomia. Ohjelmassa, jossa on useampi ikkuna käytössä, täytyy ikkunat esittää yksitellen, ei samanaikaisesti kuten MDI:ssä. Tällöin siirtyminen ikkunoiden välillä on suunniteltava etukäteen, jolloin myös käyttäjän ohjattavuus helpottuu. Kuvassa 20 on esitetty SDI-tyyppinen käyttöliittymä, joka sisältää kolme



KUVA 20 SDI-tyyppinen käyttöliittymä, jossa on kolme ikkunaa

erillistä ikkunaa. Siirtymät ikkunoiden välillä on suunniteltu siten, että a-ikkunasta voi siirtyä vain b-ikkunaan ja takaisin. B-ikkunasta voi siirtyä a-ikkunaan ja c-ikkunaan. C-ikkunasta voidaan siirtyä vain takaisin b-ikkunaan.

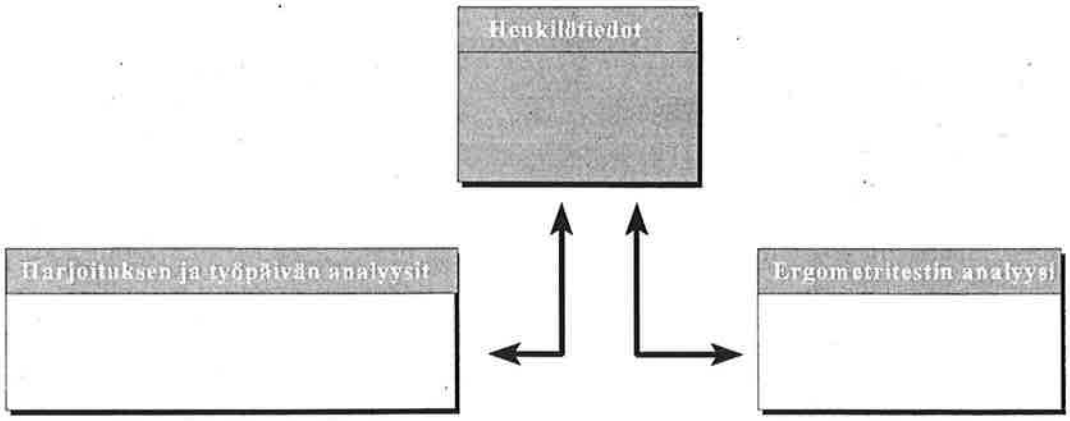
SDI-käyttöliittymän huonona puolena on se, että ohjelmassa liikkuminen tapahtuu aina samoja "polkuja" pitkin. Tämä voi kokeneen ohjelman käyttäjän mielestä olla turhauttavaa, sillä tilannehan on sama kuin että jo ennestään luetussa kirjassa pitäisi edetä sivu kerrallaan päästäkseen kirjan viimeiselle sivulle. Ohjelman suunnittelija voi tietysti rakennella erilaisia polkuja ohjelman ikkunoiden välille, mutta toisaalta se saattaa tehdä ohjelmasta monimutkaisemman käyttää.

Loppujen lopuksi valitessani käyttöliittymätyyppiä, tulin siihen tulokseen, että suurin tekijä yksinkertaisen käyttöliittymän tekemisessä ei ole käyttöliittymän tyyppi vaan sen suunnittelu. Hyvällä suunnittelulla voi MDI-käyttöliittymästä tehdä hyvin helppokäyttöisen ja vastaavasti huonolla suunnittelulla SDI-käyttöliittymästä erittäin harhauttavan ja vaikean käyttää. Niinpä tein FitWare-ohjelman käyttöliittymässä sovitteluratkaisun MDI:n ja SDI:n välillä: käyttöliittymästä tuli MDI-tyyppinen, mutta se oli käyttäjälle päin SDI-tyyppinen. MDI-käyttöliittymän hyviä puolia ovat mm. että lapsi-ikkunat pysyvät pääikkunan sisällä, eivätkä harhaile ympäri näyttöä, sekä että pääikkunan ylävalikko (menu) näkyy kaikilla ikkunoilla, joten sitä ei tarvitse erikseen piirtää (eikä ohjelmoida) jokaiselle ikkunalle. Siihen, että käyttöliittymästä tuli käyttäjälle päin SDI-tyyppinen, vaikutti ennen kaikkea vaatimusmäärittely, jossa vaadittiin, että käyttöliittymässä ei saa näkyä useita yhtäaikaisia ikkunoita, ja

määriteltiin, että käyttäjäryhmä on kokematonta tietokoneohjelmien käytössä. Lisäksi oli mainittu, että ohjelman on toiminnaltaan oltava käyttäjää opastava.

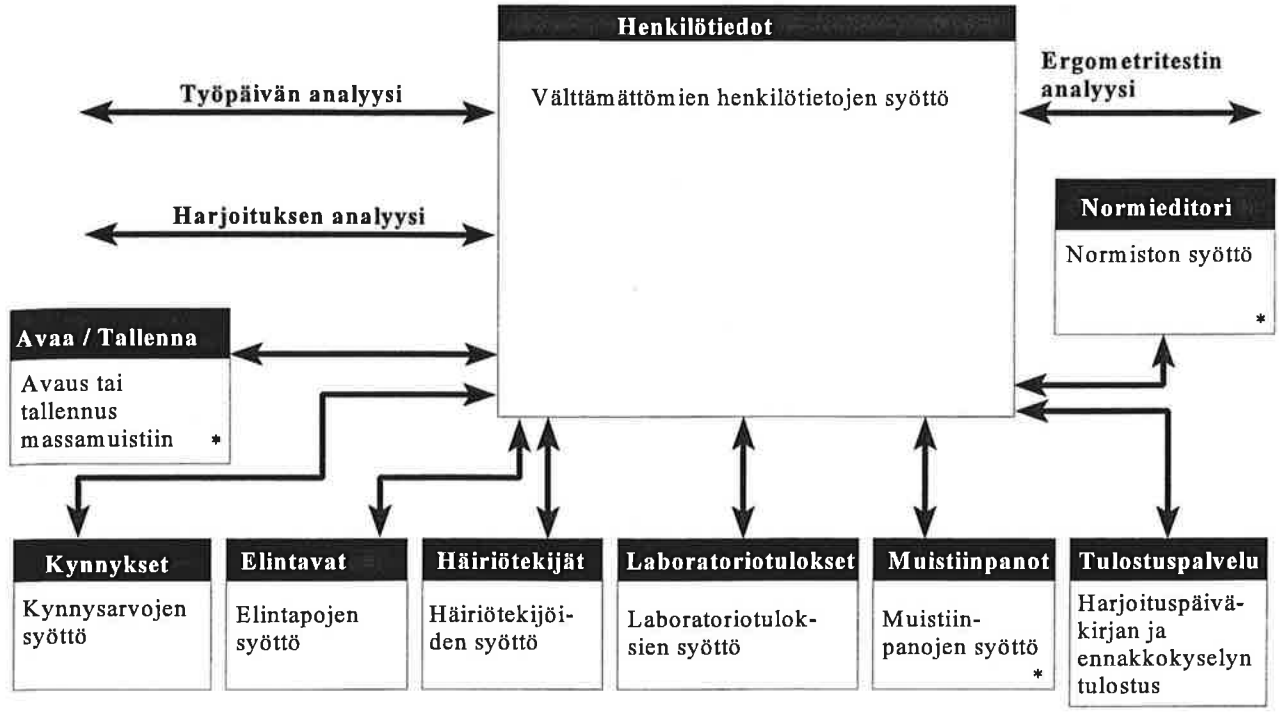
5.3.2 Käyttöliittymän ikkunointi

Käyttöliittymän ikkunointi tarkoitti käytännössä sitä, että vaatimusmäärittelyn toiminnot ryhmitelttin eri ikkunoihin. Koska käyttöliittymätyypistä tuli ulospäin SDI-tyyppinen, oli ensin määriteltävä tarvittavat ikkunat sekä niiden väliset siirtymät, eli polut, joita pitkin käyttäjä liikkuu. Koska jokainen FitWare-ohjelman analyysi tarvitsee alkutietoja, päätin sijoittaa henkilötietojen syöttöikkunan ohjelman alkuikkunaksi (ikkuna, joka ilmestyy ensimmäisenä näkyviin ohjelman käynnistyessä). Henkilötietojen syöttöikkunasta voidaan siirtyä kolmeen eri analyysiin (kuva 21). Tällä menetelmällä voidaan varmistua, että henkilötiedot syötetään ensin, ennen kuin siirrytään analyyseihin. Koska harjoituksen ja työpäivän fyysisen kuormittavuuden analyysit ovat lähes samanlaiset, niissä voidaan käyttää samaa pääikkunaa.

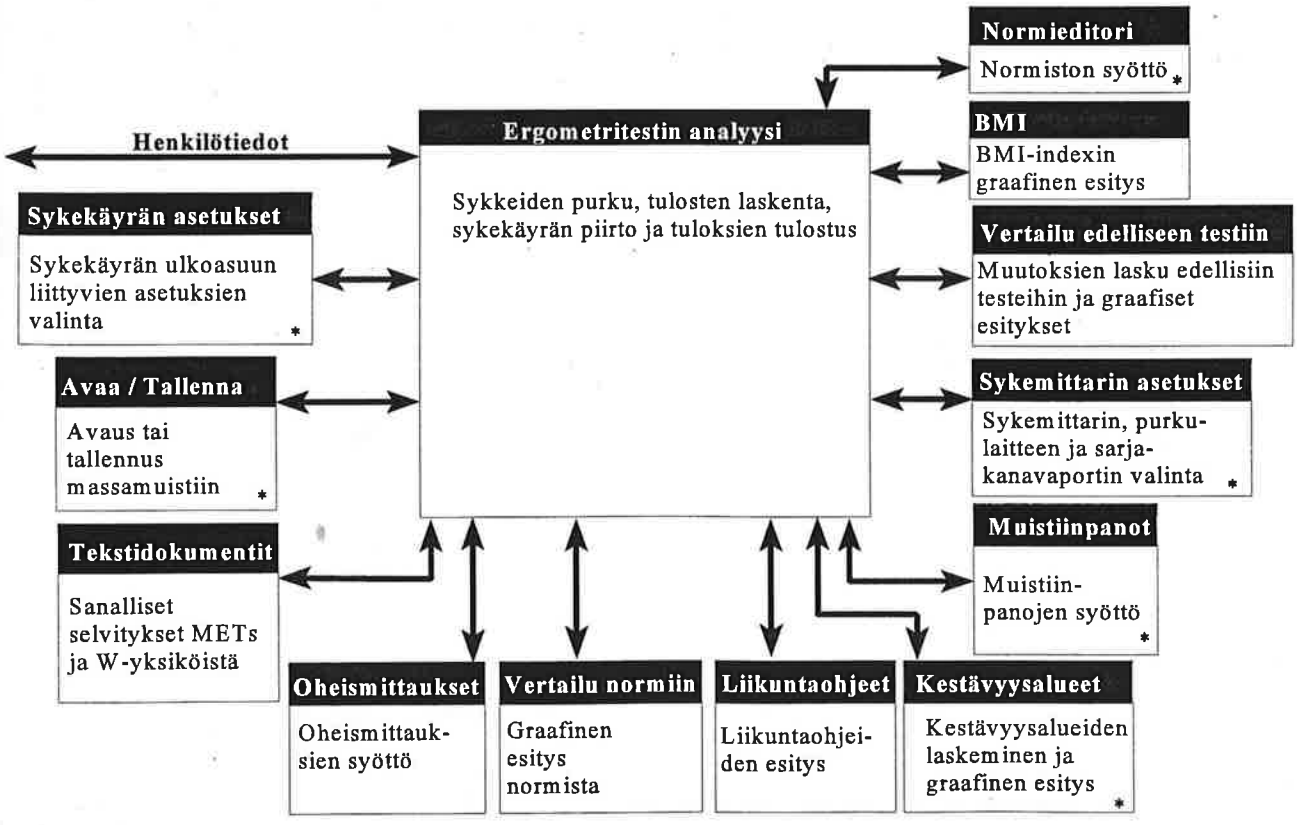


KUVA 21 FitWare-ohjelman jako kolmeen pääikkunaan ja niiden väliset siirtymät

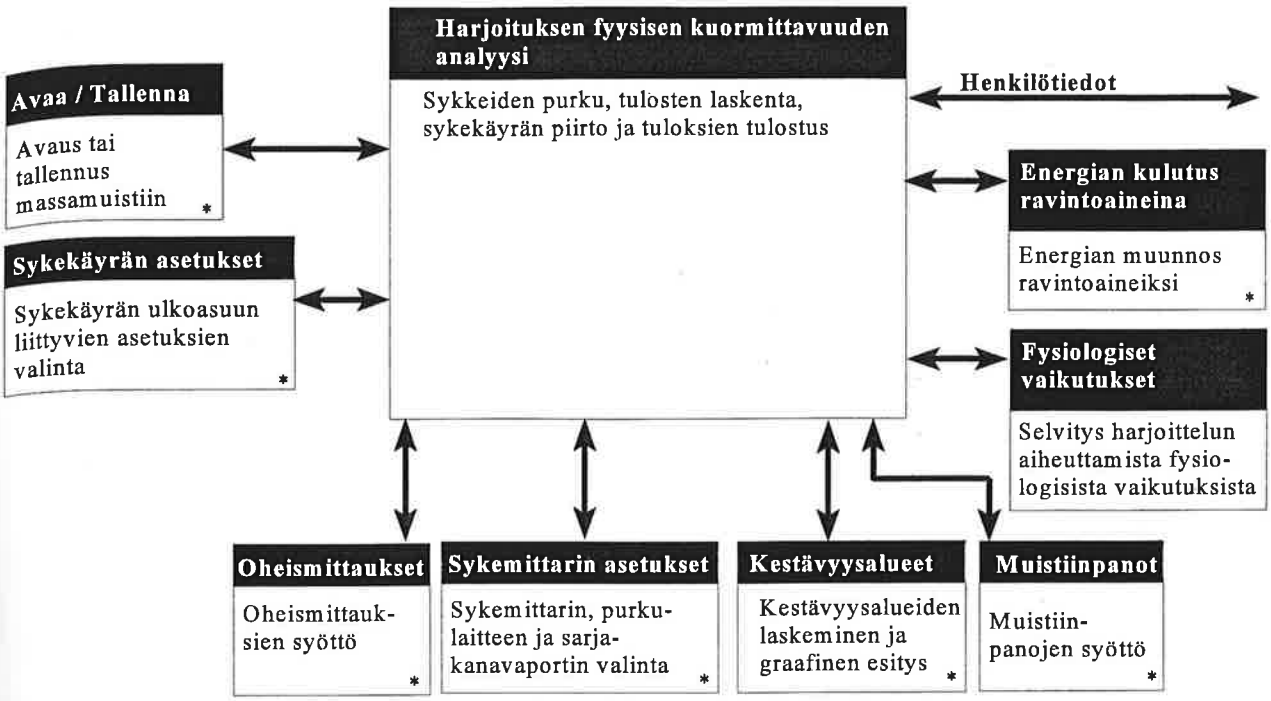
Ikkunointi eteni seuraavaksi siten, että sijoitin pääikkunoihin perustoiminnot, eli välttämättömät henkilötietojen syöttämisen henkilötietoikkunaan sekä sykkeiden purkamisen, sykekäyrät ja analyysien tulokset analyysien ikkunoihin. Näin käyttäjä voi suorittaa kaikki kolme analyysiä pelkästään käyttämällä pääikkunoita. Muut toiminnot, joita tarvitaan tarkempaan analysointiin, sijoitin erillisiin apuikkunoihin, jotka aukeavat pääsivuilta. Tällä tavoin ohjelman rakenne pysyy loogisena ja hierarkkisena kokonaisuutena käyttäjälle päin. Kuvissa 22, 23, 24 ja 25 on esitetty FitWare-ohjelman pääikkunat ja niiden alta aukeavat apuikkunat sekä niiden tehtävät.



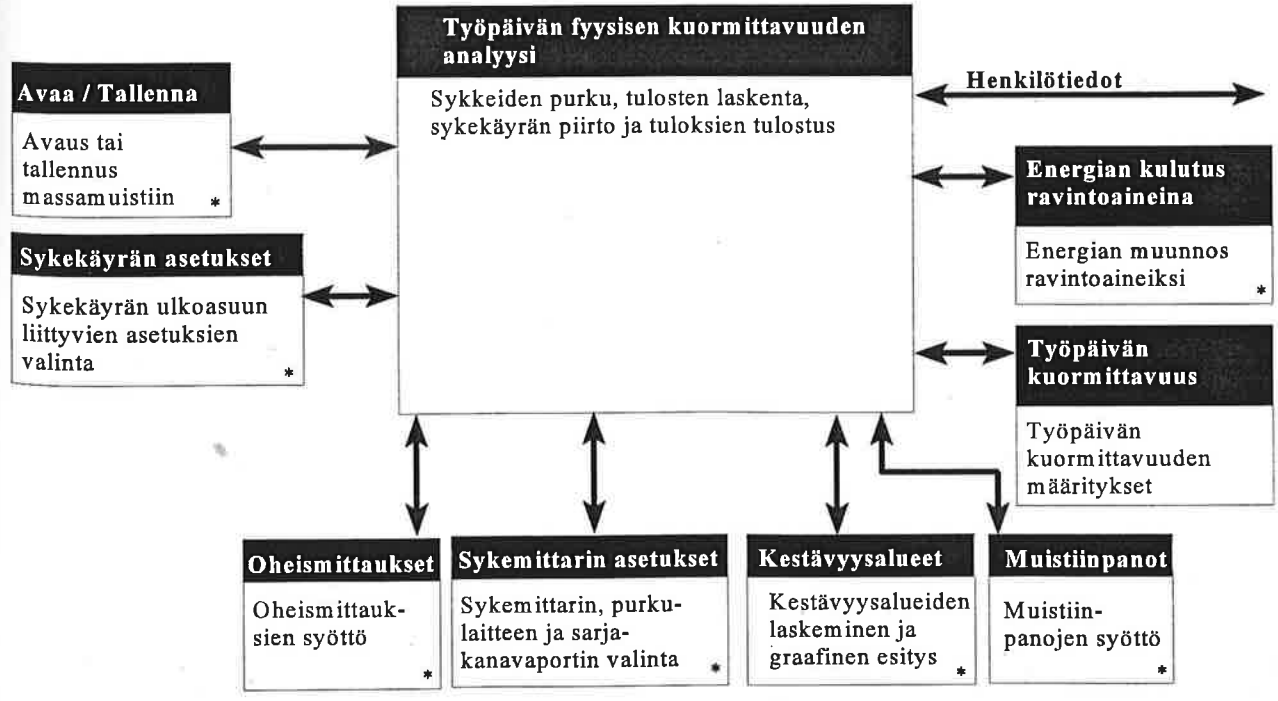
KUVA 22 Henkilötietojen pääikkuna ja sen apuikkunat sekä niiden tehtävät FitWare-ohjelmassa



KUVA 23 Ergometritestin analyysi-ikkuna ja sen apuikkunat sekä niiden tehtävät FitWare-ohjelmassa



KUVA 24 Harjoituksen fyysisen kuormittavuuden analyysi-ikkuna ja sen apuikkunat sekä niiden tehtävät FitWare-ohjelmassa.



KUVA 25 Työpäivän fyysisen kuormittavuuden analyysi-ikkuna ja sen apuikkunat sekä niiden tehtävät FitWare-ohjelmassa.

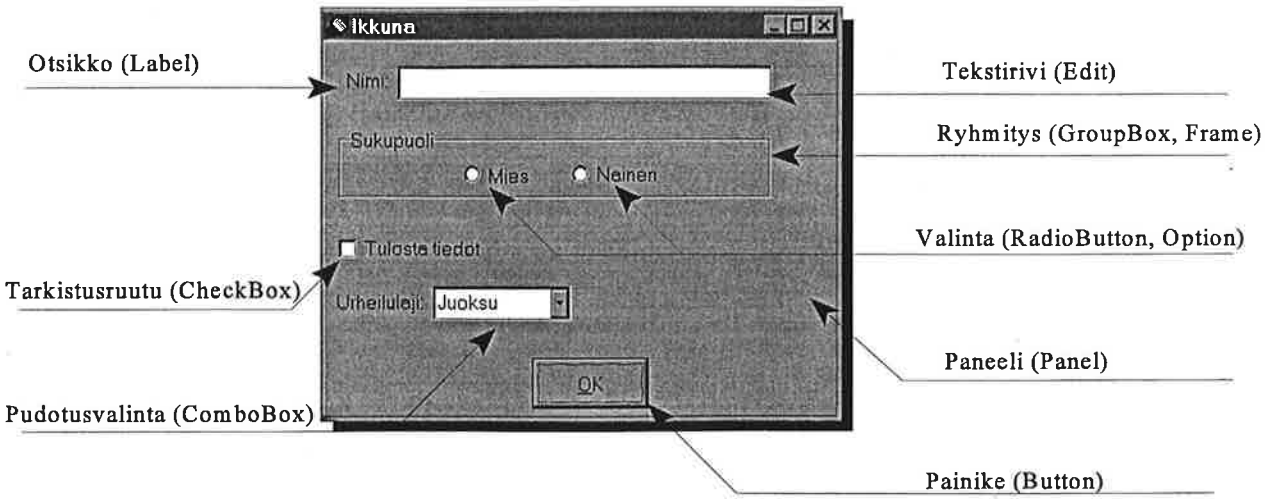
Kuvissa 22 - 25 näkyvät samannimiset apuikkunat, jotka ovat merkitty tähdellä (*) ovat samoja ikkunoita, mutta ne aukeavat vain monesta eri kohtaa. Ne erotellaan käyttäjälle käyttämällä otsakkeita. Samoin harjoituksen ja työpäivän analyysin pääikkunna on yksi ja sama ikkuna, jonka otsake vain vaihtuu analyysin mukaan. Ikkunoiden yhteiskäytöllä sain pienennettyä ohjelman kokoa, ja samalla ohjelman suorituskyky parani hieman.

Lähes kaikista apuikkunoista tuli tilasta riippuvia (modaalisia). Tämä tarkoittaa, että käyttäjä ei saa suljettua ikkunaa antamatta oikeata ohjausta, joka tavallisesti oli painikkeen painaminen. Käyttäjän kannalta nämä pakkotilanteet ovat kiusallisia silloin, kun ohjelma rajoittaa vapaata liikkumista, mutta toisaalta sillä ohjataan kokemattomia käyttäjiä. Markku Metsämäen mukaan modaalisten ikkunoiden käytössä ohjelman kerronta on usein töksähtelevää (3, s. 34), mutta henkilökohtaisesti olen sitä mieltä, että modaalisuudella saadaan käyttöliittymä toimimaan varsin keskusteletavasti, ja tähän olikin FitWare-ohjelmassa tarkoitus pyrkiä. Painikkeiden painaminen antaa käyttäjälle välittömästi palautteen ja hän oppii ohjelman käytön hyvin vähällä opiskelulla.

5.3.3 Näkyvien olioiden eli komponenttien sijoittelu

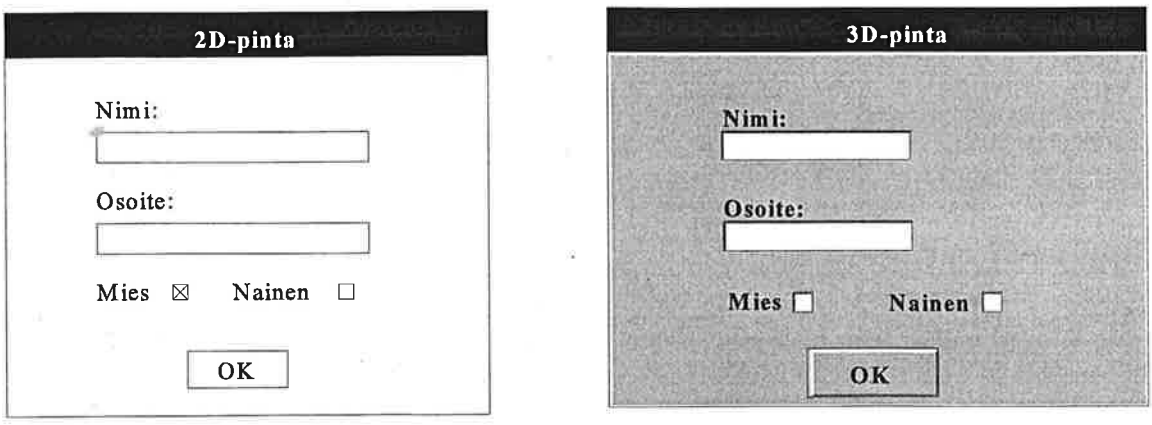
Graafisessa käyttöliittymässä ikkunoiden sisältönä ovat näkyvät oliot, joita tavallisesti ovat painikkeet, säätimet, tekstisymboolit, valintaruudut, erilaiset listat ja kuvapohjat. Oliot toimivat ikään kuin rakennuskomponentteina, joiden avulla ikkunan ulkonäkö rakennetaan ja joita ohjelmoimalla ohjelman toiminnot toteutetaan. Näkyvistä olioista käytetäänkin usein pelkästään nimitystä komponentti. Komponentit toimivat myös rajapintana, jonka kautta käyttäjä "keskustelee" ohjelman kanssa. Komponentteja voidaan ohjata ohjelmakoodilla ja ne voivat olla joko näkyviä tai näkymättömiä. Kuvassa 26 on esitetty Windows-sovelluksen yleisimmät komponentit.

Näkyvät oliot voidaan sijoittaa ikkunoihin ja muuttaa niiden ulkonäköä haluamallaan tavalla - ohjelmakoodia ei tarvita. Tämä piirre ei ole ainoastaan tyypillistä Visual Basic-ohjelmalle, vaan lähes kaikille Windows-ohjelmointikielille. Näkyville olioille voidaan antaa ominaisarvoja suunnitteluvaiheessa, joista ehkä tärkein on olion nimi. Olion nimi kannattaa muuttaa suunnitteluvaiheessa sen toimintaa kuuvaavaksi. Esim. jos painikkeen tehtävänä on sulkea ikkuna, sen nimeksi kannattaa antaa vaikkapa "sulje_ikkuna_nappi". Oikealla nimeämisellä saadaan olion paikallistamisesta ja havainnollistamisesta helpompaa ohjelmakoodissa.



KUVA 26 Windows-sovelluksen yleisimmät näkyvät komponentit ja niiden nimitykset

Aloitin olioiden sijoittelun ikkunoihin välittömästi sen jälkeen kun ikkunointi oli pääpiirteissään suunniteltu. Tässä suunnitteluvaiheessa, kuten ikkunoinnissakin, käytettiin hyvin paljon katselmuksia avuksi, koska käyttöliittymän ulkoasua ei oltu määritelty kovinkaan tarkasti vaatimusmäärittelyssä. Olioiden sijoittelussa ja ulkonäössä pyrin mahdollisemmanselkeään ja yksiselitteiseen ratkaisuun. Lähtökohtana olioiden sijoitteluun oli, että näytön minimiresoluutio olisi 640*480 pikseliä ja pienin värimäärä 16 väriä. Näytön yleisilmeeksi valittiin kolmiulotteisuus eli tilan tunnun omaava tyyli. Ensimmäiset Windows-ohjelmat olivat kaksiulotteisia eli litteitä, mutta koska viimeaikoina lähes kaikki ohjelmat ovat olleet kolmiulotteisia, ei FitWare-ohjelman tyylissä haluttu poiketa yleisesti totutusta mallista. Kuvassa 27 on esitetty kaksi- ja kolmiulotteinen käyttöliittymäpinta.

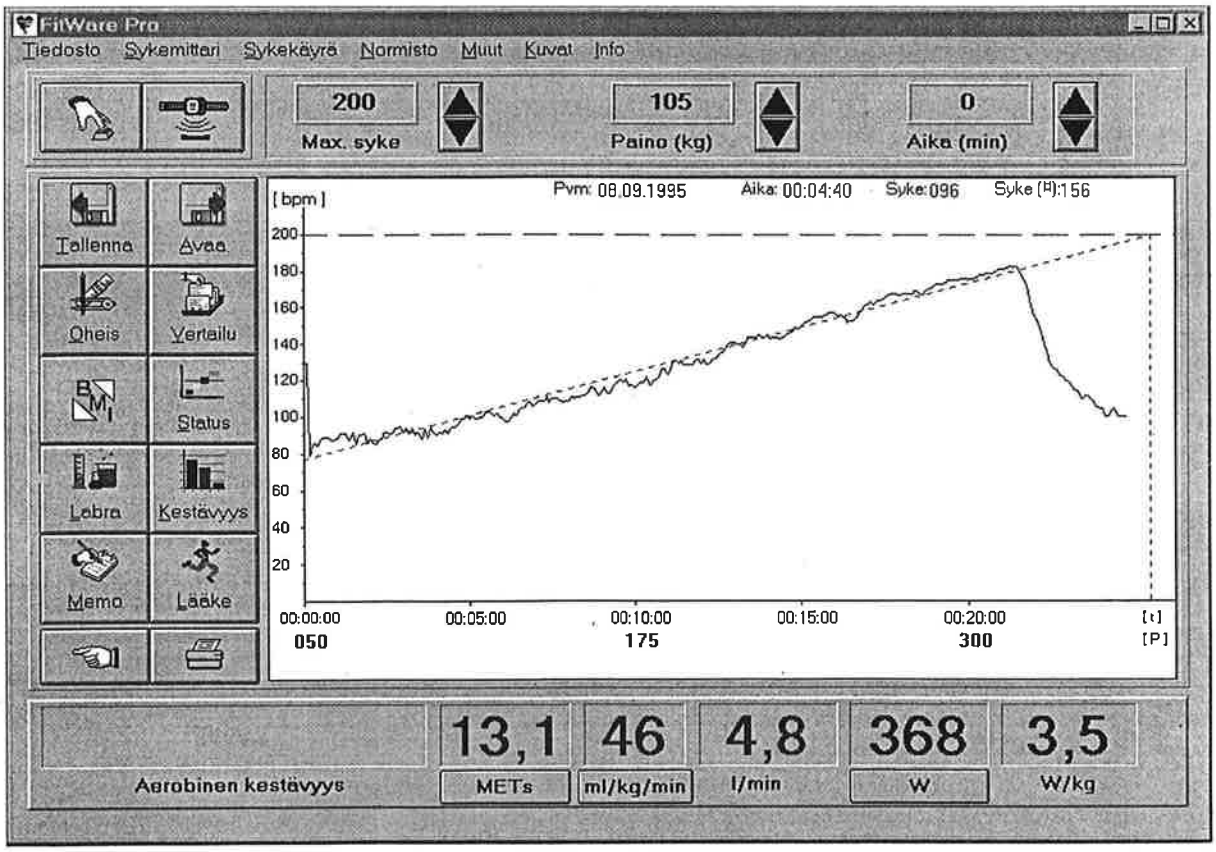


KUVA 27 Kaksiulotteinen ja kolmiulotteinen käyttöliittymäpinta Windows-käyttöjärjestelmässä

Olioiden sijoittelussa ikkunoihin oli tärkeätä säilyttää sama tyyli ja logiikka jokaisessa ikkunassa, jotta ohjelman oppimiskynnys pysyisi matalana. Jos jokainen ikkuna olisi tyylillään erilainen, myös ohjelman yhtenäisyys kärsisi ja ohjelmasta tulisi häiritsevä. FitWare-ohjelman tunnusomaisiksi piirteiksi tulikin isot kuvakepainikkeet, jotka sijaitsivat ikkunan sivulla. Jos ikkunassa oli tuloksia, sijoitin yleensä ne ikkunan alaosaan. Sykekäyrät asettelin ikkunoiden keskelle mahdollisimman suureksi.

Koska FitWare-ohjelman yksityiskohtainen dokumentointi on salaista, käsittelen tästä edespäin esimerkkinä vain yhtä FitWare-ohjelman ikkunaa ja sen tiettyjä toimintoja. Tämä menettely on välttämätöntä myös sen takia, että kaikkien FitWare-ohjelman ikkunoiden ja toimintojen esittäminen tässä työssä veisi aivan liian paljon aikaa ja kasvattaisi sivujen lukumäärää liiaksi. Käytän esimerkkinä ohjelman ergometritesti-ikkunaa ja sen sykekäyrätoimintoja.

Kuvassa 28 on ergometritestianalyysin ikkuna. Ikkunan vasemmassa reunassa on painikkeita, joista voi siirtyä apuikkunoihin. Ikkunan alaosassa on tekstiruudut tuloksille ja ruudun keskellä on tila sykekäyrälle. Ikkunan vasemmassa ylälaidassa on painikkeet sykkeiden purkamista varten ja ylhäällä olevia tekstiruutuja käytetään tulosten korjaamiseen.



KUVA 28 Ergometritestianalyysin ikkuna FitWare-ohjelmassa

Kuvan 28 ikkunan ylävalikosta (menu) löytyvät seuraavat toiminnot:

Tiedosto	Avaa, Tallenna, Tulosta, Poistu
Sykemittari	Lue sykemittari, Sykemittausasetukset, Kellon asetukset, Asetukset
Sykekäyrä	Piilota sykesuunta, Päivitä sykekäyrä, Asetukset
Normisto	Normi 1...Normi 10, Asetukset
Muut	Yhteenveto, Virheen suodatus, Ergometrin ohjaus
Kuvat	Sydän, Verenkiertojärjestelmä, Kohonnut verenpaine
Info.	

Ylävalikon toiminnoista osa on sama kuin ikkunan painikkeillakin - kaikkia painikkeiden toimintoja ei kuitenkaan löydy ylävalikosta. Ylävalikon toiminto 'kuvat' on lisätty FitWare-ohjelmaan juuri tätä kirjoitettaessa, joten sitä ei ole mainittu vaatimusmäärittelyissä. Kuva-toiminnolla tarkoitetaan, että nimikettä vastaava kuva tulee näkyviin, esim. sydän-toiminto avaa kuvan sydämestä. Kuvilla saadaan lisättyä ohjelman havainnollisuutta ja ne toimivat testaajan apuvälineenä.

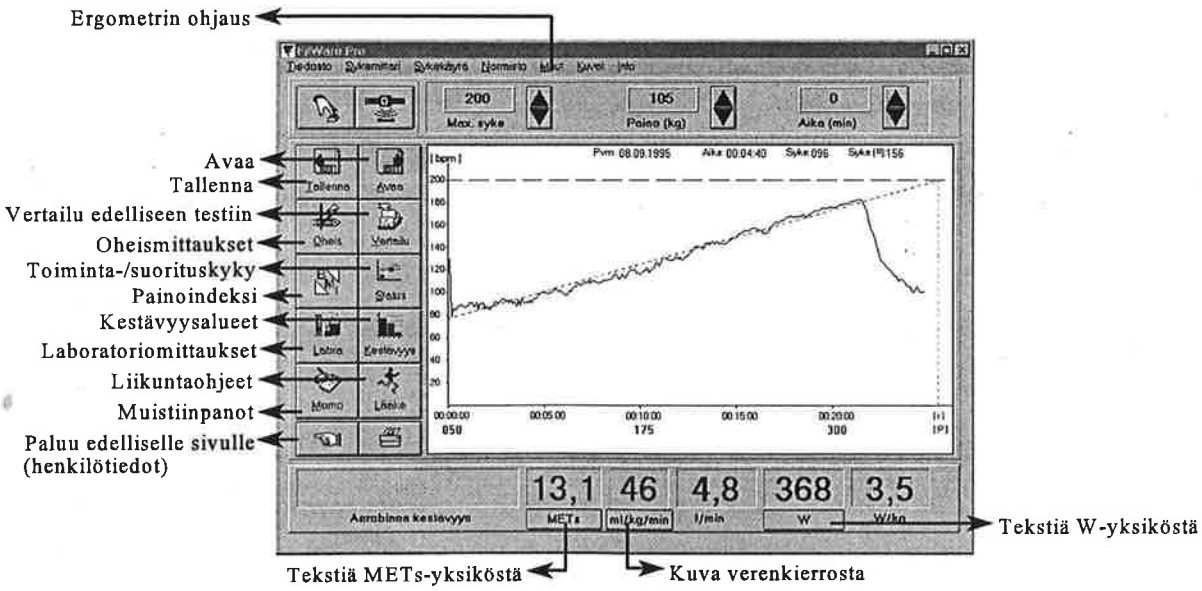
Ergometritestianalyysin ikkunassa käytetyt komponentit ovat yleisiä Windows-komponentteja, mutta sykekäyrän piirtoon käytetystä kuva-komponentista voisi mainita muutaman seikan. Kuva-komponenttiin (PictureBox) voidaan avata valmiita BMP-muodossa olevia kuvia tai siihen voidaan pirtää ajon aikaisesti. Komponentin ominaisuuksista voidaan valita päivittääkö ohjelma automaattisesti kuvaa vai hoitaako ohjelmoitsija itse sen ohjelmakoodissa. Automaattinen kuvan päivitys on kätevä, mutta jos kuvan sisältö muuttuu tai sisältö on iso, toiminta on erittäin hidasta. Käytin automaattista päivitystä ergometritestin sykekäyrässä, koska sisältä muuttuu harvoin, mutta sen sijaan työn ja harjoituksen sykekäyrissä käytin "manuaalista" päivitystä, koska sykekäyrää oli pystyttävä lähentämään (zoomaus) ja sen jälkeen vierittämään vaakatasossa. Automaattisella päivityksellä toiminta olisi ollut erittäin hidasta.

Kuva-komponentin metodeihin kuuluva normaalien perustoimintojen lisäksi hiiren liikkeisiin reagoivat toiminnot (esim. MouseDown, MouseUp), joten sen soveltuminen sykekäyrän piirtämiseen ja etenkin hiiri-analysointiin oli erittäin hyvä.

FitWare-ohjelmassa käytettävät painikkeiden kuvakkeet ovat suurilta osin itse tehtyjä, mutta joissain painikkeissa käytettiin myös valmiskuvakkeita. Kuvakkeet eivät syntyneet lopullisiksi ensimmäisellä yrittämällä, vaan niiden muodot ja värit muuttuivat työn edetessä.

5.4 Toimintojen ryhmittely

FitWare-ohjelman ergometritesti-ikkunan toiminnoista jakaantuvat ergometritestin tulokseen liittyviin toimintoihin ja ikkunasta aukeaviin apuikkunoihin. Kuvasta 29 nähdään, että suurin osa ikkunan painikkeista on tarkoitettu apuikkunoiden aukaisua varten. Sykkeiden syötön jälkeen ohjelma laskee automaattisesti tuloksen, mutta käyttäjä voi tämän jälkeen vaikuttaa tulokseen joko muuttamalla henkilön maksimisykettä tai painoa. Maksimisykettä voi muuttaa myös lisäämällä arvioitua 'jaksamisaikaa', eli lisäämällä testin pituutta ajallisesti testattavan tuntemuksen mukaan. Lisäksi käyttäjä voi 'maalata' sykekäyrältä alueen jonka mukaan tulokset lasketaan. Kuvassa 30 on esitetty eri mahdollisuudet tuloksien muuttamiseksi. Muita toimintoja ergometri-ikkunaan liittyen ovat testituloksien tulostus, sykemittareiden ja sykekäyrän asetukset, virheiden suodatus, normiston valinta ja asetukset sekä yhteenveto testistä. Näiden lisäksi sykekäyrällä näkyy jatkuvasti hiiren kursoria vastaava aika ja syke sekä aikaa vastaava syke.



KUVA 29 Ergometritesti-ikkunasta aukeavat apuikkunat

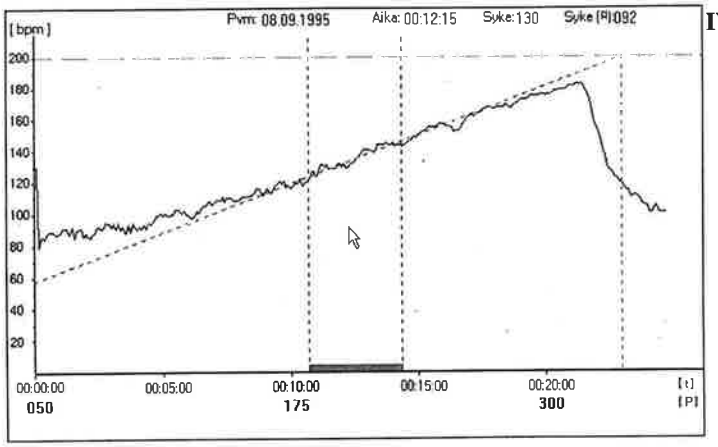
Keskityn tästä eteenpäin tarkastelemaan ainoastaan toimintoja, jotka tapahtuvat käyttäjän liikuttaessa hiirtä sykekäyrän alueella. Nämä toiminnot toimivat esimerkkinä toimintojen suunnittelusta ja toteuttamisesta. Sykekäyrällä näkyvät toiminnot ovat siis jatkuva hiiren kursoria vastaavan ajan ja sykkeen sekä aikaa vastaavan sykkeen näyttäminen sekä mahdollinen maalaustoiminto. Maalaustoiminto toimii seuraavasti: maalaustoiminto aktivoituu,

kun käyttäjä napsauttaa hiirtä sykekäyrän kohdalla. Käyttäjä valitsee maalauksen aloituskoh dan painamalla hiiren vasemman napin pohjaan. Hiirtä liikutamalla, ja samalla painamalla hiiren nappia, hän maalaa sykekäyrän aluetta. Kun hiiren vasen nappi lasketaan irti maalaus loppuu ja maalauskohtaa vastaava tulos ja sykesuuntaviivat lasketaan välittömästi. Maalaus poistuu painamalla uudestaan hiiren vasenta nappia. Toimintaa voi havainnollistaa kuvan 31 mukaisesti. Sykekäyrä on kuva-komponentti, jolla on eri metodeja. Edellä mainitut toiminnot voidaan yhdistää olion eri metodeihin.

I Käyttäjä voi muuttaa maksimisykettä

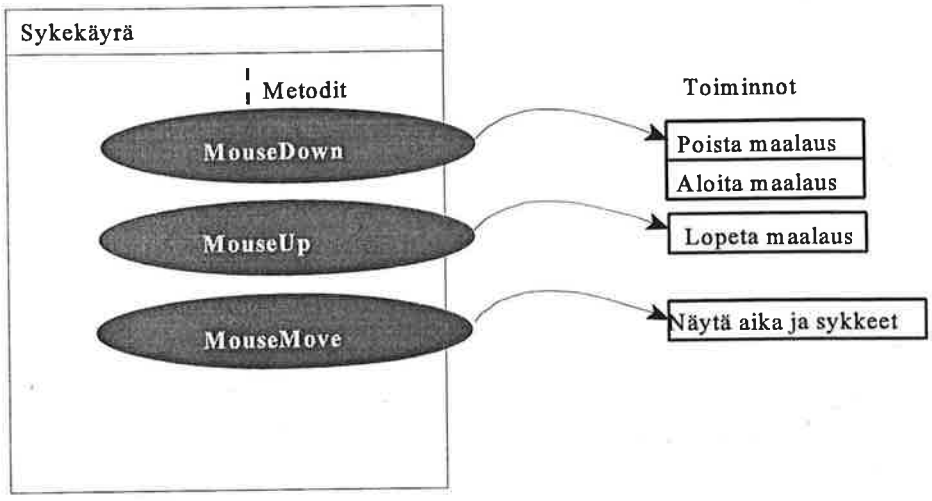
II Käyttäjä voi muuttaa painoa

III Käyttäjä voi lisätä testin pituutta



IV Käyttäjä voi maalta haluamansa alueen sykekäyrältä

KUVA 30 Eri tapoja, miten käyttäjä voi korjata ergometritestin tulosta FitWare-ohjelmassa



KUVA 31 Sykekäyrä-olion metodit ja niitä vastaavat toiminnot

6 FITWARE-OHJELMAN YKSIKKÖSUUNNITTELU JA KOODAUS

FitWare-ohjelman mallinnuksessa ja toteutuksessa käytettiin siis Visual Basic 3.0 Pro-ohjelmaa. Suurin osa ohjelman koodista sijoittui komponenttien tapahtumakäsittelijöihin. Tämä tarkoittaa sitä, että esim. painikkeella on tapahtuma 'Click' ja jos siihen sijoitetaan koodia, se aktivoituu, kun tapahtuma 'Click' tulee voimaan, eli kun tässä tapauksessa painiketta painetaan. Kaikkia toimintoja ei kuitenkaan voi sijoittaa tapahtumakäsittelijöihin, tällöin on tehtävä omia proseduureja ja funktioita, jotka voivat olla myös globaaleja.

Osa toiminnoista oli niin yksinkertaisesti toteutettavissa, että erillistä yksikkösuunnittelua ei tarvittu. Jotkin toiminnot oli kuitenkin pakko suunnitella etukäteen ennen ohjelmointia. Ohjelmointia helpotti se, että Visual Basic- ja yleensäkin Windows-ohjelmointi on tapahtumapohjaista eikä peräkkäisyyksiä synny kovin paljon. Siten koodin hajautuminen yksiköihin tapahtui ikään kuin automaattisesti, eikä pitkiä koodisivuja päässyt syntymään.

Visual Basic antaa hyvin vapaat kädet ohjelmoitsijalle, joka ei aina ollut hyväksi. Esimerkiksi muuttujien esittelyä ei Visual Basicissä tarvitse tehdä. Tämä tuotti aluksi ongelmia, koska globaalit ja paikalliset muuttujat menivät silloin tällöin sekaisin. Lopussa ohjelmointikieleen alkoi kuitenkin tottua ja ongelmia syntyi enää hyvin harvoin. Kaiken kaikkiaan Visual Basicin käyttö oli helppoa ja siihen oli helppo päästä sisään, mutta tietynlaista kurinalaisuutta jäin ohjelmalta kaipaamaan.

6.1 Ohjelmointi Visual Basicillä

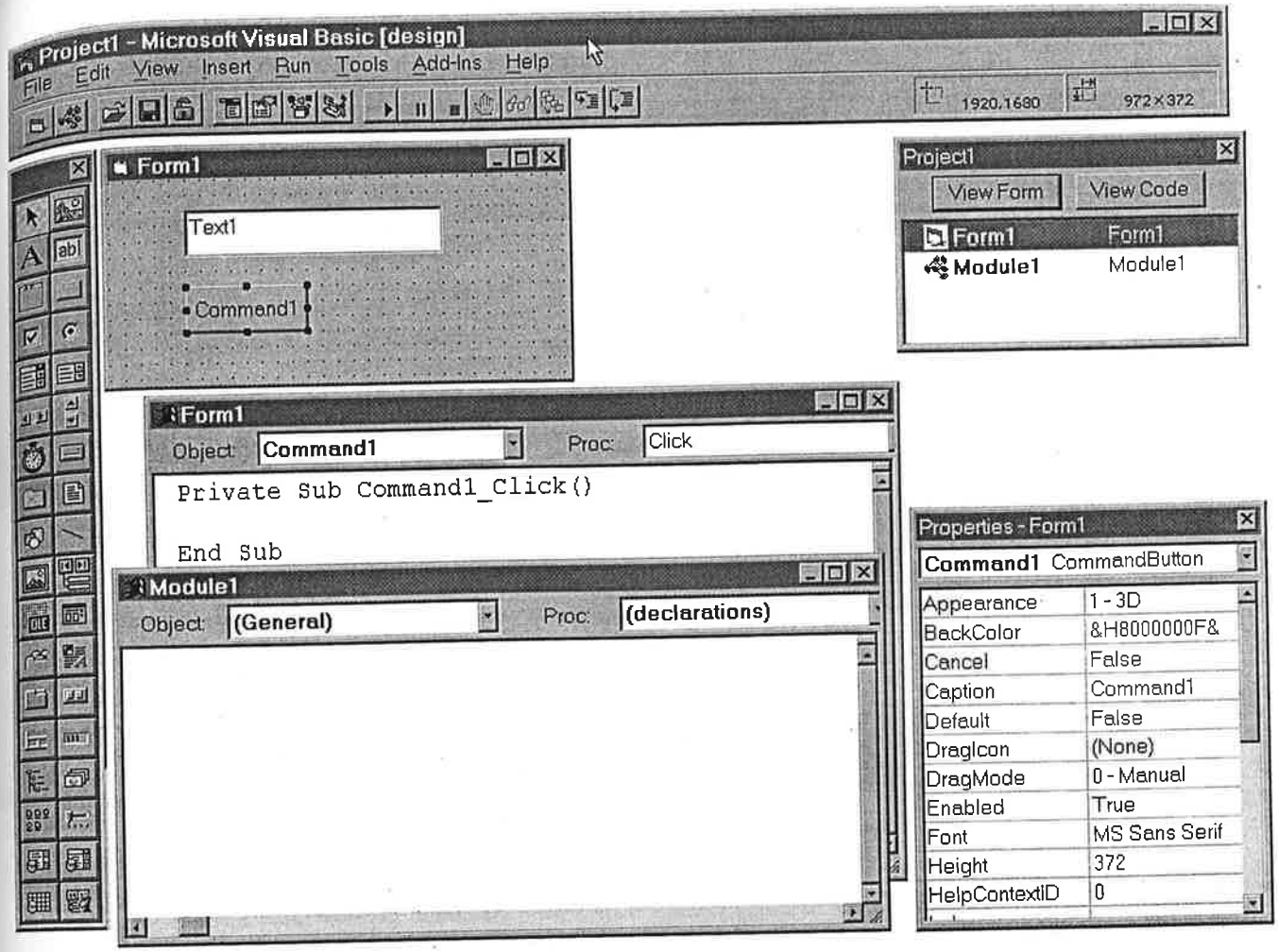
Visual Basic-sovelluskehittimen ohjelmointikielenä on Basic. Basic ei tarjoa yhtä laajoja mahdollisuuksia kuin esim. C- tai Pascal-kieli, mutta yllättävän moni ongelma on ratkaistavissa sen avulla. Periaatteessa Visual Basicillä voi ratkaista minkä tahansa ongelman käyttämällä Windows:n API-ohjelmointirajapintaa (Application Program Interface) ja sen rutiineja avuksi. Näiden käyttö on kuitenkin paljon monimutkaisempaa kuin Basic-kielen. Samalla myös ohjelman siirrettävyys kärsii, etenkin 16- ja 32-bittisten käyttöjärjestelmien välillä.

Visual Basicin perusideana on, että käyttäjä sijoittaa komponenttipaletista komponentit ikkunaan ja tämän jälkeen liittää komponentin tiettyyn tapahtumaan koodi-ikkunassa haluamansa koodin. Visual Basicin ikkunaa kutsutaan lomakkeeksi (form) ja se sisältää ikkunan

ja sen komponentit sekä niihin ohjelmoidut koodit. Ikkunoita voi olla useita, mutta muuttuja ja funktiot eivät näy eri ikkunoiden välillä, ts. ikkunasta ei voi kutsua toisen ikkunan funktiota tai lukea toisen ikkunan muuttujia. Jos halutaan käyttää globaaleja funktioita tai muuttujia, jotka näkyvät siis kaikille ikkunoille, Visual Basicissä käytetään 'moduulia' (module). Moduulilla ei ole ikkunaa, johon komponentteja voisi asetella, vaan ainoastaan koodi-ikkuna, johon voi määritellä muuttujia ja proseduureja sekä funktioita. Jokainen ohjelma on Visual Basicissä projekti (project). Kun ohjelma tallennetaan, tallennetaan lomakkeet, moduulit sekä projekti. Projektitiedosto pitää sisällään tiedon mitä lomakkeita ja moduuleita on käytetty ja missä ne sijaitsevat. Lisäksi tiedostossa on määritelty, mitä VBX-kirjastoja on käytetty. VBX-kirjastot sisältävät komponentteja sekä niihin liittyviä funktioita.

Kuvassa 32 näkyy Visual Basicin käyttöliittymä. Visual Basicin pääikkuna näkyy ylhäällä, jossa sijaitsevat ohjelman perustoiminnot (avaa, tallenna, asetukset jne.). Vasemmassa reunassa on komponenttipaletti (toolbox), josta käyttäjä raahaa komponentit lomakkeelle. Keskellä näkyy suunniteltava ikkuna (form1), johon on sijoitettu tekstiruutu (text1) ja painike (command1). Tämän alapuolella on koodi-ikkuna, jossa näkyy aktivoitua komponenttia vastaava kohta (command1). Koodi-ikkunan oikeassa reunassa on liukuvalitsin, josta voidaan valita haluttu tapahtuma. Kuvassa näkyvä tapahtuma on 'Click'. Koodi-ikkunan alapuolelle on sijoitettu moduuli-ikkuna (module1), jossa tehtävät määrittelyt näkyvät kaikkialla. Oikeassa yläreunassa on projekti-ikkuna, josta näkyy projektin sisältävä lomakkeet ja moduulit. Oikeassa alareunassa on aktivoitua komponenttia, kuvassa painike, vastaavat ominaisuudet (properties), joita voidaan muuttaa suunnitteluvaiheessa.

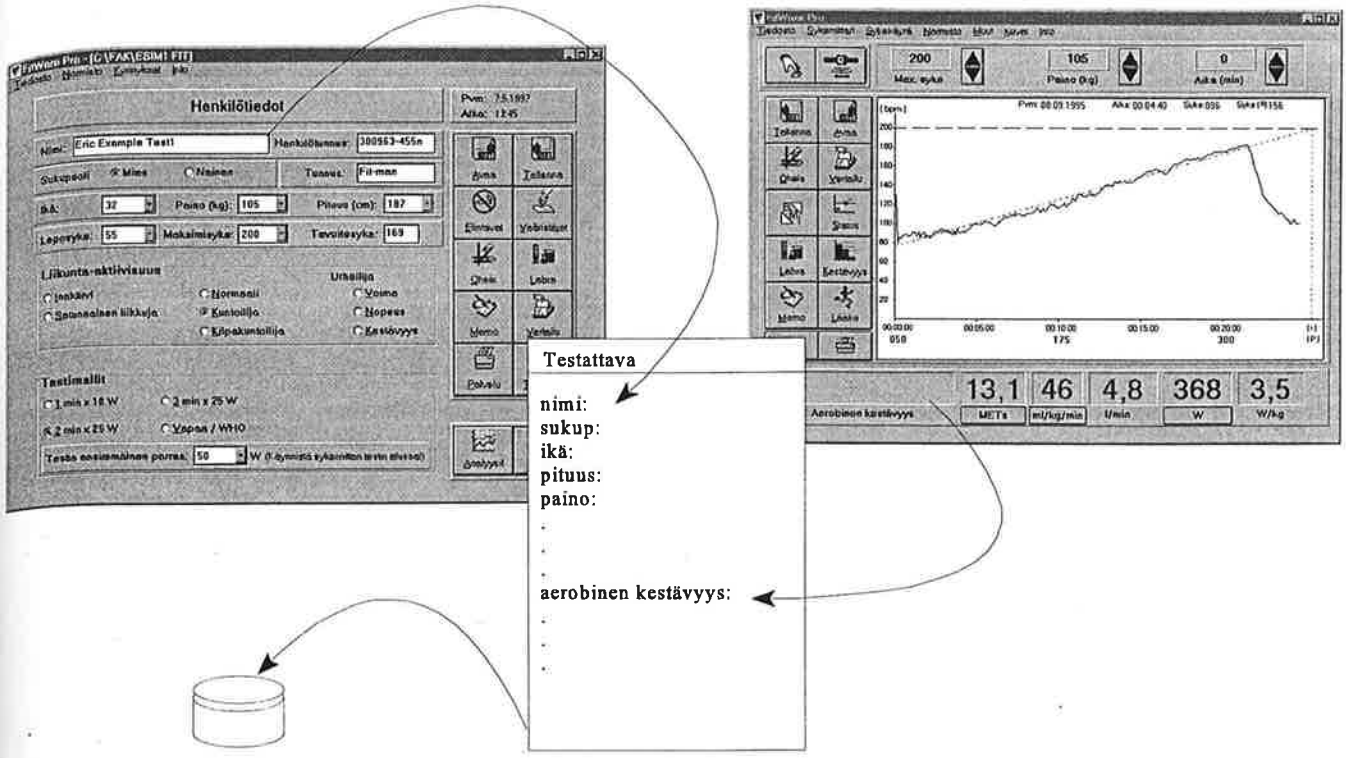
Visual Basicillä tehtyä ohjelmaa voi ajaa Visual Basicillä ilman käännöksiä ja linkittämistä - Visual Basic ei tee konekielikäännöstä. Ohjelman käynnistäminen on nopeaa ja sen vuoksi toimintojen testaus on näppärää. Haittapuolena on se, että valmiin sovelluksen toiminta ei välttämättä ole nopeaa, mutta sovellutuksissa, missä ei käytetä raskasta laskentaa, eroa konekielikäännettyihin ohjelmiin on vaikea huomata. Visual Basicillä tehty sovellutus tarvitsee toimiakseen aputiedostoja: VBRUN-kirjaston, GSW-grafiikkarajapinnan sekä käytetyt VBX-kirjastot. Käännetyt ohjelmat eivät ole kooltaan kovin isoja, mutta tarvittava levytila kasvaa juuri näiden tiedostojen takia.



KUVA 32 Visual Basicin käyttöliittymä

6.2 FitWare-ohjelman ohjelmointi

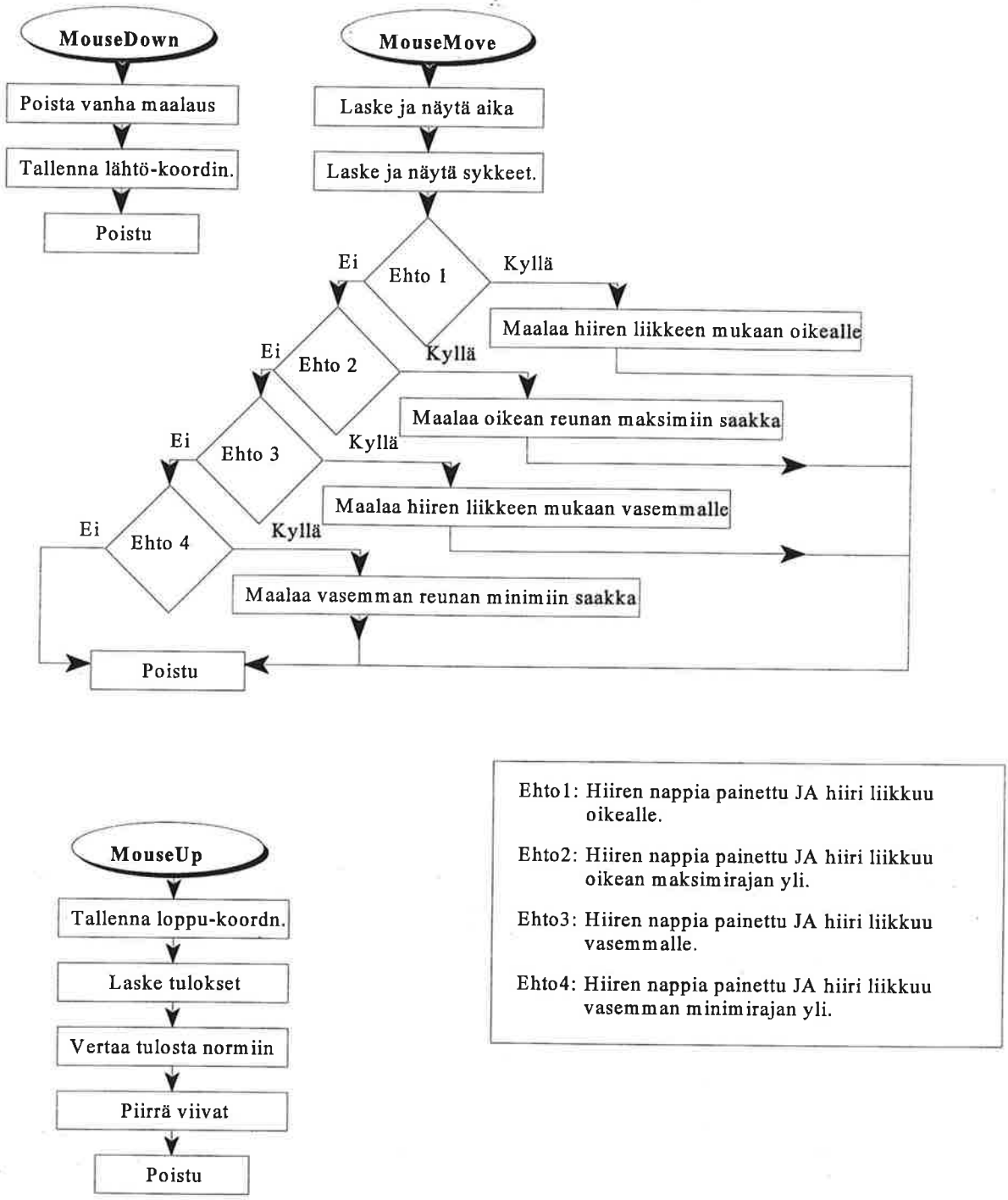
Suuri osa ohjelman toiminnoista oli jonkinlaisia ohjauksia, kuten siirtymiset ikkunoiden välillä, joten yksikkösuunnittelua ei aina tarvittu. Ohjelman toimintoja pystyi ohjelmoimaan ja testaamaan ikkuna kerrallaan, mikä nopeutti ohjelman kokoamista. Ennen ohjelmoinnin aloittamista oli kuitenkin suunniteltava ohjelman tietorakenteet ja etenkin muuttujien ja funktioiden näkyvyys. Koska FitWare-ohjelmassa käsiteltävä tieto on aina testattavaan liittyvää tietoa, päätin tehdä testattavan tiedoista oman tietotyypin, joka näkyisi kaikissa ikkunoissa. Tein testattavasta tavallaan olion, joka sisälsi muuttujia, kuten nimi, ikä, sukupuoli, pituus jne. Näin testattavan tiedot pysyvät jatkuvasti ajan tasalla ohjelman suoritusaikana ja niiden muuttaminen on helppoa. Kuva 33 havainnollistaa toimintaa. Olio luodaan ohjelman käynnistysvaiheessa. Käyttäjän syöttäessä henkilötietoja, ne siirtyvät suoraan olion ominaisuuksiksi. Kun ergometritestin tulokset on laskettu, myös tulokset siirtyvät saman olion ominaisuuksiksi. Käyttäjän tallentaessa, tallennetaan ainoastaan ko. olio.



KUVA 33 FitWare-ohjelmassa käytetään testattavan tiedon keräämiseksi erillistä oliota, joka sisältää kaikki tarvittavat muuttujat liittyen testattavaan

Olioita olisi helppo luoda useampi ajon aikana, jolloin käyttäjä voisi käsitellä useamman testattavan tietoja yhtä aikaa. Jotta ohjelma pysyisi yksinkertaisena käyttäjälle päin, oli kuitenkin järkevää käsitellä vain yhtä testattavaa kerrallaan.

Tämän jälkeen ikkunoiden itsenäinen ohjelmointi oli helppo aloittaa. Kuten jo mainitsinkin, suuri osa ohjelmoinnista oli erilaisiin tapahtumiin reagoivien toimintojen ohjelmointia, sekä tietojen sijoitteluun liittyvää ohjelmointia. Toimintoja, joita ei voinut suoraan ohjelmoida, mallinsin vuokaavioiden avulla. Esimerkkinä vuokaavion käytöstä käytän luvussa 5.4 esitellyn toiminnon (hiiren liikkuminen ergometritesti-ikkunan sykekäyrällä) mallintamista. Toiminnot on jaettava kolmen eri tapahtuman kesken: hiiren nappi alhaalla (MouseDown), hiiren nappi ylhäällä (MouseUp) ja hiiri liikkuu (MouseMove). Toiminnot voidaan esittää kuvan 34 mukaisesti. Hiiren painiketta painamalla maalaus poistetaan ja tallennetaan uuden maalauskohdan aloituskoordinaatit. Hiirtä liikuttaessa, lasketaan ja näytetään hiiren kursoria vastaava aika ja sykkeet. Tämän jälkeen tarkastetaan mihin suuntaan hiiri on maalaamassa. Kun hiiren painike lasketaan irti, talletetaan maalauskohdan lopetuskoordinaatit, lasketaan tulokset, verrataan tulosta normiin ja piirretään lopuksi sykesuuntaviivat ym. sykekäyrässä tarvittavat viivat. Vuokaavion perusteella tehty ohjelmakoodi löytyy liitteestä 6.



KUVA 34 Vuokaavioesimerkki ergometri-ikkunan sykekäyrän hiiritoiminnoista FitWare-ohjelmassa

Ohjelman kokoaminen eteni juuri edellä mainitulla tavalla, eli kohdat, joissa yksikkösuunnittelu oli tarpeen, suoritin ennen koodausta vuokaavio-mallinnuksen, ja ne toiminnot, jotka olivat yksinkertaisia, ohjelmoin ilman vuokaavioita.

FitWare-ohjelman yksikkö- ja integrointitestauksen suoritin itse, mutta testaukset siitä eteenpäin olivat työnantajan ja hänen testausryhmänsä vastuulla. Testaukset etenivät yleensä niin, että kun olin saanut tietyn prototyypin osajärjestelmän toimimaan, toimitin sen työnantajan testattavaksi. Koska ohjelma ei ollut kovin riippuvainen osajärjestelmistään, pystyin

testauksien välissä kokoamaan toista osajärjestelmää. Koska testaus itsessään on hyvin laaja osa ohjelmistokehitystä, en käsittely FitWare-ohjelman testausta tässä työssä sen tarkemmin.

7 FITWARE-OHJELMAN KÄYTTÖÖNOTTO JA TULEVAISUUDEN NÄKYMÄT

FitWare-ohjelmaa on otettu käyttöön vähitellen. Tämä tarkoittaa sitä, että työnantaja on ottanut valmistuneet prototyypit välittömästi käyttöön työnantajan ja toivotut muutokset ja virheiden korjaukset on otettu huomioon seuraavien prototyyppien valmistuksessa. Ohjelma on tätä kirjoitettaessa (kevät 1997) edennyt versioon 1.2. Ensimmäistä 1.0-versiota pidettiin lähes valmiina tuotteena, mutta etenkin Polar Electro Oy:n julkaisemat uudet kellot pakottivat ohjelman jatkokehittelyn versioon 1.1. Jälkeenpäin ajateltuna versiot 1.0 ja 1.1 olivat vielä puhtaita prototyyppejä, ja vasta versio 1.2 on valmis tuote. Versiossa 1.2 on ohjelman toimintavarmuutta paranneltu ja se sisältää myös uusia toimintoja, kuten opetuskäyttöön tarkoitetut kuvat.

Ohjelman alkukehittelystä asti FitWare-ohjelmasta on tehty kolme eri versiota: Basic, Exe ja Pro. FitWare Basic -versio pitää sisällään ainoastaan ergometritestaukseen ja analysointiin liittyvät toiminnot, Exe-versio sisältää edellisen lisäksi harjoitusvaikutusanalyysin ja Pro-versio tietenkin kaikki toiminnot. FitWare-ohjelmasta on tulossa myös kouluympäristöön tarkoitettu versio, jota myydään oppilaitoksille valmiina järjestelmänä yhdessä polkupyöräergometrin ja sykemittareiden kanssa mahdollisimman huokeaan hintaan.

FitWare-ohjelma ei pidä sisällään ohje-tiedostoa, mikä on tietenkin ohjelman puute. Toisaalta ohjelman mukana toimitetaan ohjekirjanen, ja tämän lisäksi ohjelman käyttäjille järjestetään koulutustilaisuuksia. Tällä hetkellä ohjelmaa on käytössä parissakymmenessä paikassa ympäri Suomea, mm. puolustusvoimilla ja työterveyslaitoksilla, eikä käyttäjiltä ei ole tullut valituksia siitä, että ohjelman käyttöä olisi vaikea oppia tai että ohjelman käyttö itsessään olisi vaikeata. Sen sijaan paljon ideoita ja parannusehdotuksia käyttäjiltä on tullut, mikä on tietenkin hyväksi ohjelman jatkokehittelyä varten.

Fit-Test Oy:llä on tarkoitus viedä ohjelmaa myös ulkomaille, ja tämän takia FitWare-ohjelma on käännetty tällä hetkellä englanniksi, ruotsiksi ja saksaksi. Yhteydenottoja Englantiin,

Ruotsiin, Saksaan sekä Australiaan on alustavasti jo tehty, ja vastaanotto on ollut erittäin lämmintä.

FitWare-ohjelman kehitys ei tietenkään pysähdy nykyiseen versioon, vaan Fit-Test Oy:n tarkoituksena on kehittää FitWare-ohjelmaa entistä paremmaksi. Tulevista ideoista ja parannuksista en ymmärrettävistä syistä voi tässä työssä mainita, mutta uskon, että ohjelmaan tulee vielä monta uutta oivallusta ja että ohjelman pedagoginen puoli paranee entisestään. Uskon myös, että ohjelman toteuttaminen jollakin muulla ohjelmointikielellä tai sovelluskehittimellä, kuin Visual Basicillä, kuuluu tulevaisuuden suunnitelmiin.

8 MITÄ OPIN TYÖSTÄ?

FitWare-ohjelman kehittely opetti minulle valtavasti uusia asioita, koska sain olla työssä mukana kaikkien kehitysvaiheiden aikana. Oli erittäin opettavaista nähdä koko ohjelmistokehityksen kaari ja sen eri työvaiheiden ongelmat sekä niiden ratkaisut. Ohjelmistotekniikan teoriassa on painotettu esitutkimuksen, dokumentoinnin ja testauksen osuutta ohjelmistokehityksessä, mutta vasta tämän työn jälkeen todella huomasin, miten tärkeitä asioita nämä ovat ohjelmakehityksen kannalta. FitWare-ohjelmiston kehitys poikkesi hieman normaalista ohjelmistokehityksestä kuitenkin siinä, että ohjelman lopullinen käyttäjä oli jatkuvasti läsnä ohjelman kehityksessä. Näin puutteellisia vaatimusmäärittelyjä pystyttiin paikkaamaan. Tämä oli mielestäni hyväksi tämänkaltaiselle ohjelmistolle, sillä itse pidän ohjelmaa eräänlaisena asiantuntijajärjestelmänä, jonka tunnuspiirteenä on lopullista käyttäjää mahdollisimman pitkälle palveleva toiminta. Tämän toteuttaminen pelkän vaatimusmäärittelyn perusteella olisi ollut mahdotonta.

Koska työnantajalla ei ollut kokemusta tietotekniikasta, oli projektin aikana opeteltava keskustelemaan työnantajan ja myös muiden asiantuntijoiden kanssa ohjelmistoteknisistä asioista "yhteisellä kielellä". Tämän takia FitWare-ohjelman kehittely oli aloitettava prototyyppien rakentelulla. Näin työnantaja pystyi tarkentamaan vaatimuksiaan ja samalla ottamaan kantaa ohjelman ulkonäköön. Tässä vaiheessa työskentelyni oli eräänlaista tulkkausta työnantajan ja tietokoneen välillä. Työnantaja saattoi sanoa, että "tähän ikkunaan pitää saada nappi, jota painamalla pääsee takaisin edelliseen ikkunaan", ja sen jälkeen minä "kerroin" sen tietokoneelle ohjelmoimalla vastaavat toiminnot. Tämän tapainen työskentelytapa johti jatkuvien muutosten ja parannuksien tekoon (ja saattoi olla joskus myös henkisesti raskasta),

mutta lopullisesta tuotteesta tuli ainakin mahdollisemman pitkälle käyttäjän toivomuksien mukainen.

Ohjelman kehittely antoi myös kokemusta sellaisista asioista, joita ei kirjoja lukemalla voi oppia. Esimerkiksi eri työvaiheiden kustannuksien ja toteutusaikojen arviointi ilman minkäänlaista alan kokemusta on mahdoton tehtävä. Samoin myös eri työtehtävien vaikutuksia toisiinsa on vaikea ennakoida ilman kokemusta. Tietyt asiat ohjelman tekemisessä on opeteltava myös kantapään kautta. Niinpä kaikki virheet työn aikana ovat hyväksi opiksi tulevaisuutta varten - samoja virheitä ei tule tehtyä enää uudelleen tai ne osaa ainakin ennakoida.

FitWaren kehittelytyön myötä opin myös Windows-ohjelmoinnin perusteet. Tätä pidän erittäin hyödyllisenä oppina, koska lähes kaikki ohjelmat tehdään nykyään Windows-ympäristöön, ja ilman tätä työtä en todennäköisesti olisi perehtynyt Windows-ohjelmointiin näin perusteellisesti. Visual Basicin käytöstä on ohjelmistotyön jälkeen tullut erittäin rutiininomaista, ja sen myötä olen tutustunut myös muiden Windows-sovelluskehittimien ja -ohjelmointikielten käyttöön. Työn jälkeen on helppo arvioida, mihin tehtäviin eri ohjelmistotyökalut parhaiten soveltuvat. Tämä on arvokasta tietoa silloin, kun halutaan minimoida ohjelmistokehityksen ajankäyttö ja kustannukset.

Yksi tärkeä aihealue, johon jouduin perehtymään, oli graafisen käyttöliittymän suunnittelu. Moni saattaa väheksyä käyttöliittymän merkitystä ohjelman kannalta, mutta FitWare-ohjelman kehitystyö osoitti minulle sen olevan yksi ohjelmiston tärkeimmistä tekijöistä. Käyttöliittymä antaa ensivaikutelman ohjelmasta. Käyttäjää ei kiinnosta lainkaan se, kuinka ohjelman toiminnot on toteutettu tai millä ohjelmointikielellä ohjelma on kasattu, vaan se, että ohjelma toiminnoiltaan ja etenkin ulkonäöltään vastaa käyttäjän tarpeita. On tärkeä muistaa, että lähes jokaisesta kaupallisesta ohjelmasta löytyy kilpaileva versio, ja silloin käyttäjä saattaa valita ohjelman pelkästään käyttöliittymän perusteella. Käyttöliittymällä voidaan myös suoraan vaikuttaa ohjelman käyttömukavuuteen ja poistaa käyttäjän virheitä. Hyvän graafisen käyttöliittymän toteuttaminen on myös yksi niistä asioista, joita ei opi muuta kuin kokemuksen kautta.

Nykyaikaisen tietokoneohjelman kehittelyyn ei mielestäni riitä enää ainoastaan pelkkä ohjelmistotekniikan osaaminen. Koska ohjelmien visuaalisuus ja tietosisältö kasvavat jatku-

vasti, tarvitaan ohjelman kehitystyöhön avuksi myös tukihenkilöiksi media-alan ammattilaisia. Ohjelman tietosisällön ja etenkin esitystavan suunnittelussa voisi olla hyvin mukana vaikkapa graafisia suunnittelijoita ja kielen huollon ammattilaisia. FitWare-ohjelman valmistuttua huomasin, että moni tiedon esittämiseen liittyvä asia olisi ehkä voitu tehdä ohjelmassa paremmin käyttämällä alan asiantuntijoita.

FitWare-ohjelman kehitystyö antoi minulle kokemusta myös tiimityöskentelystä. Tiimityössä tiimin on pystyttävä luomaan oikea ilmapiiri, jotta työn lopputuloksesta tulisi mahdollisimman onnistunut. Mielestäni tulosta tuottava ryhmä ei vietä aikaa keskenään ainoastaan työaikana, vaan myös vapaa-aikana. Näin ryhmän jäsenet tutustuvat toisiinsa paremmin ja ryhmätyöstä tulee antoisampaa. Vapautunut keskusteleminen ja mielipiteiden vaihtaminen ryhmätyössä on tärkeää työn onnistumisen kannalta. Samalla syntyy myös uusia ideoita. Työskentely FitWare-ohjelman parissa opetti, että ryhmän jäsenten on pystyttävä ottamaan vastuuta asioista ja vastaavasti luottamaan toisiin ryhmän jäseniin. Ryhmän jäsen saattaa laiminlyödä tehtävänsä, mutta toisaalta hänelle saattaa tulla pelko siitä, että toiset ryhmän jäsenet eivät tee tai eivät osaa tehdä omia tehtäviään. Tällaisia tilanteita syntyy ryhmän sisällä juuri silloin, kun se vielä ei ole tarpeeksi hyvin "hitsautunut" yhteen.

Lopuksi haluaisin painottaa suunnittelun ja dokumentoinnin osuutta ohjelmistokehityksessä. Nämä ovat juuri niitä asioista, jotka opitaan usein kantapään kautta. Monen asian kimppuun syöksytään ilman minkäänlaista suunnitelmaa, ja kun ongelma on lopuksi jotenkin saatu ratkaistua, siitä ei jätetä minkäänlaista dokumenttia. Usein käytäntöä perustellaan ajan säästönä, mutta tosiasiassa asia on juuri päinvastainen. Huolella suunniteltu työ ja sen dokumentointi säästävät kokonaisaikaa, etenkin testausvaiheessa. Ilman dokumentointia vian etsintä on erittäin työlästä, ja usein käykin niin, että vikaa ei viitsitä edes paikantaa, vaan se ehkäistään jollakin muulla toimella. Tästä juontuu myös se yleinen mielikuva, että ohjelmistotyö on pelkkää koodausta. Täytyy myöntää, että FitWare-ohjelman kehitystyössä sorruin edellä mainittuihin vääriin työskentelytapoihin melko usein, mutta mielestäni olen nyt läksyni tältä osin oppinut.

LÄHTEET

1 Ojalehto, Sauli. Ohjelmistotekniikka. Helsinki: Valtion painatuskeskus, 1990. 357 s. ISBN 951-37-0050-X.

2 Metsämäki, Markku. Graafinen käyttöliittymä - GUI. Helsinki: Painatuskeskus Oy, 1995. 125 s. ISBN 951-37-1511-6.

3 Ohjelmistotekniikan luennot. Muistiinpanot ja kurssimateriaali, syksy 1995. Mikkeli: Mikkelin ammattikorkeakoulu.

4 Ahlberg, Janne. Windows-ohjelmointi: johdatus ja perusteet. Jyväskylä: Suomen ATK-kustannus Oy, 1996. 235 s. ISBN 951-762-364-X.

5 Ihme, Tuomas & Pesonen, Pekka. Oliokeskeiset menetelmät sulautettujen ohjelmistojen kehittämisessä. Helsinki: TEKES julkaisu 33/92, 1992. 91 s.

6 Becks, Ari. Delphi - sovelluksentekijän opas. Jyväskylä: Suomen ATK-kustannus Oy, 1995. 624 s. ISBN 951-762-289-9.

7 Taari, Timo. Asiantuntijajärjestelmän rakentaminen. Ylivieska, 1990. 29 s. ISBN 952-9540-01-9.

8 McKinney, Bruce. Tehokäyttäjän opas: Visual Basic. Jyväskylä: Suomen ATK-kustannus Oy, 1996. 618 s. ISBN 951-762-435-2.

LIITE 1

Liikuntalääke-ikkunan oletustekstit

FitWare-ohjelmassa

Harjoittelu-alue	Harjoittelun tärkeys	Sykealue	Liikunnan kesto	Liikunnan useus	Liikuntalaji
Perus-kestävyys	*Erittäin suositeltavaa*	*50% - 70% max sykkeestä*	*Vähintään 30 min*	*3 - 5 kertaa viikossa*	*Reipas kävely, pyöräily, uinti, hiihto*
Vauhti-kestävyys	*Suositeltavaa*	*70% - 90% max sykkeestä*	*15 - 60 min*	*3 - 5 kertaa viikossa*	*Rasittavampi kävely, pyöräily, uinti, hiihto, hölkkä*
Maksimi-kestävyys	*Suositeltavaa varauksin*	*90% - 100% max sykkeestä*	*5 - 30 min*	*Enintään 2 kertaa viikossa*	*Kovatehoinen kestävyys- / intervalli-harjoittelu*

LIITE 2

BORG-indeksiä vastaavat sanalliset määrittelyt

BORG-indeksi:	**Määrittely:**
6	
7	*Hyvin, hyvin kevyt*
8	
9	*Hyvin kevyt*
10	
11	*Kevyt*
12	
13	*Kohtalaisen raskas*
14	
15	*Raskas*
16	
17	*Hyvin raskas*
18	
19	*Hyvin, hyvin raskas*
20	

BORG-indeksiä käytetään FitWare-ohjelmassa sekä BORG-mittauksessa että työpäivän ja harjoituksen koetun kuormituksen mittauksessa.

LIITE 3

Energia-ikkunan elintarvikkeet ja niiden energiasisällöt FitWare-ohjelmassa

Elintarvike:	**Energiasisältö:**	
Donitsi	*1445*	*kJ / kpl*
Jäätelötötterö	*930*	*kJ / kpl*
Kevytmaito	*190*	*kJ / kpl*
Emmental-juusto	*1595*	*kJ / kpl*
Kala (hauki)	*339*	*kJ / kpl*
Suklaapatukka	*2234*	*kJ / 100 g*
III-olut	*142*	*kJ / 100 g*
Ranskalaiset	*1250*	*kJ / 100 g*
Salami	*1958*	*kJ / 100 g*
Peruna	*300*	*kJ / 100 g*
Omena	*153*	*kJ / 100 g*
Vaalea leipä	*901*	*kJ / 100 g*
Kinkku	*385*	*kJ / 100 g*
Hampurilainen	*1075*	*kJ / 100 g*
Broileri	*800*	*kJ / 100 g*

-1 gramma sisältää 38 kJ (9 kcal) rasvaa, 17 kJ (4 kcal) hiilihydraatteja ja 17 kJ (4 kcal) proteiinia.

-1 kcal = 4,25 kJ

LIITE 4

FitWare-ohjelmassa tarvittavat henkilötiedot 1 (2)

	Tieto:		Esitysmuoto:
Perustiedot:	*Nimi*		*max. 50 kirjainta*
	Henkilötunnus		*11 merkkiä*
	Sukupuoli		*mies tai nainen*
	Ikä		*5 - 80 vuotta*
	Paino		*10 - 200 kg*
	Pituus		*90 - 240 cm*
	Leposyke		*30 - 120 bpm*
	Maksimisyke		*100 - 250 bpm*
	Testin tavoitesyke		*85 % max sykkeestä*
	Tunnus (esim. seura)		*max. 15 merkkiä*
	Liikunta-aktiivisuus	*vaihtoehdot:*	*Inaktiivi*
			Satunnainen liikkuja
			Normaali
			Kuntoilija
			Kilpakuntoilija
			Voimaurheilija
			Kestävyysurheilija
			Nopeusurheilija
Elintavat:	*Oma tuntemus suorituskyvystä verrattuna samanikäiseen väestöön*	*Vaihtoehdot:*	*Heikko*
			Välttävä
			Keskitaso
			Hyvä
			Erinomainen
	Oma tuntemus suorituskyvystä verrattuna edelliseen testiin	*Vaihtoehdot samat kuin yllä*	
	Tupakointi	*Vaihtoehdot:*	*En tupakoi*
			Lopettanut (vuosi)
			Tupakoin (kuinka usein)
	Kahvi	*Vaihtoehdot:*	*En juo kahvia*
			Juon kahvia (paljonko)
	Alkoholi	*Vaihtoehdot:*	*En käytä alkoholia*
			Käytän alkoholia
			(Kuinka usein)
	Lääkitys:		*max 100 kirjainta*
	Sydän-, veri- ja hengityselinsairaudet		*max 100 kirjainta*
	Rintakipua	*Vaihtoehdot:*	*Ei*
			Levossa
			Rasituksessa
	Työmatkat	*Vaihtoehdot:*	*kuljen töihin kävellen*
			" pyörällä
			" autolla
			matka km ?
	Työn fyysinen kuormittavuus	*Vaihtoehdot:*	*Toimistotyö*
			Kevyt ruumiillinen työ
			Raskas ruumiillinen työ

LIITE 4

FitWare-ohjelmassa tarvittavat henkilötiedot 2 (2)

	Tieto:		Esitysmuoto:
	Viikottainen liikunta-aktiivisuus	*Vaihtoehdot:*	*Ei lainkaan* *Joskus* *Kerran viikossa* *2-3 kertaa viikossa* *4-5 kertaa viikossa* *Useammin*
	Liikuntalajit		*max 50 kirjainta*
	Kilpaurheilutausta		*Vaihtoehdot: Kyllä tai ei*
	Kilpaurheilulajit		*max 50 kirjainta*
Laboratoriomittaukset:	*Lepoverenpaine (sys. ja dias.)*		*kokonaisluku*
	Valkosolut		*luku muotoa 12,3*
	Punasolut		"
	Ferritiini		"
	Hematokriitti		"
	Hemoglobiini		"
	MCV		"
	MCH		"
	MCHC		"
	LASKO		"
	Veren glukoosi		"
	Totaali kolesteroli		"
	Virtsan valkuainen		"
	Virtsan glukoosi		"
	PEF		"
	FVC		"
	HDL-kolesteroli		"
	HDL-COL-suhde		"
	LDL		"
	ASAT		"
	ALAT		"
	Gamma GT		"
	Kreatiini		"
	BMI		"
	Rasvaprosentti		"
	Maksimaalinen hapenottokyky		
	- ml/kg/min		"
	- l/min		*luku muotoa 1,23*
	- METs		*luku muotoa 12,3*
	Teho/paino-suhde		"
	Maksimaalinen teho		"
	FEV 1		"
	(FVC/FEV1)%		"

LIITE 5

FitWare-ohjelman tärkeimmät tulosteet 1 (8)

TESTIPALAUTE

Testipvm: *1.2.1997* Klo: *12.00*
Nimi: *Erkki Esimerkki* Ikä: *30* Paino: *75* Pituus: *180* Sukupuoli: *Mies*

Ergometritestin mittaustulokset

Testimalli:	*10 W x 1 min*		Aloitusvastus: *50 W*	
Aerobinen kestävyys:	*Erinomainen*		Normisto: *Shvartz&Reinbold*	
METs:	*18,7*			
VO_{2max}:	*65 ml/kg/min*			
Maksimihapenkulutus:	*4,6 l/min*			
Maksimisyke:	*200*			
Maksimityöteho:	*349 W*			
Maksimityöteho:	*5,0 W/kg*			
BMI:	*24 (Ihanteellinen paino)*			
Kestävyysalueet:	*PK 46%;*	*VK 33%;*	*MAX 0%*	*testin kokonaisajasta*

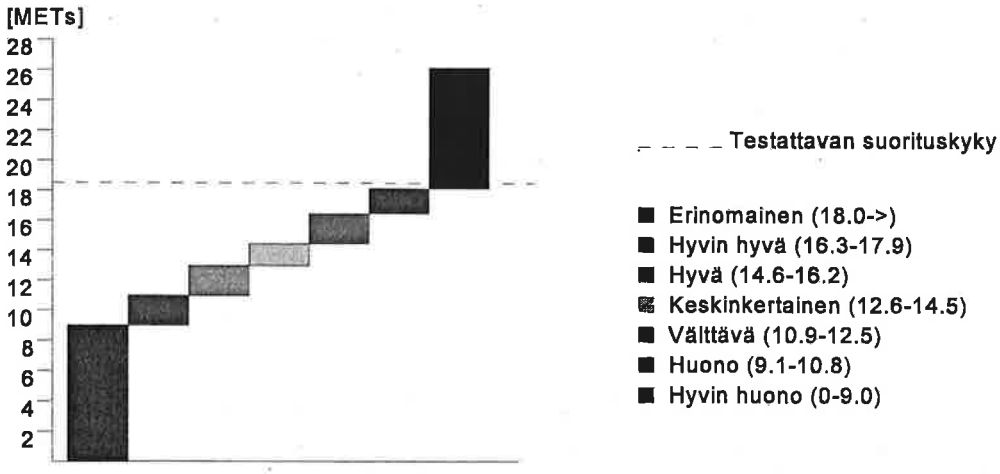
Harjoitteluohje (Harjoituskertoja 3-7/vko kestävyyden eri alueilla vaihdellen)

	Sykealue	Kesto	Useus
Peruskestävyys	*100-140*	*Vähintään 30 min*	*3-5 kertaa/vko*
Vauhtikestävyys	*140-180*	*15-60 min*	*3-5 kertaa/vko*
Maksimikestävyys	*180-200*	*5-30 min*	*Enintään 2 kertaa/vko*

Muut mittaustulokset

	Systolinen	Diastolinen
Lepoverenpaine:	*80*	*100*
Borg (testin lopussa):	*450*	

Toiminta- / suorituskyky



Kommentti:

LIITE 5

FitWare-ohjelman tärkeimmät tulosteet 2 (8)

LIIKUNTAHARJOITTELUPÄIVÄKIRJA

Viikko nro: Harjoitusviikko nro:

Harjoitusviikon tavoitteet:

Harjoituksen määrä: ..... kertaa viikossa
Syketaso harjoituksessa: .....-..... kertaa minuutissa
Alkuverryttelyn kesto: ..... min
Harjoittelujakson kesto: ..... min
Loppuverryttelyn kesto: ..... min

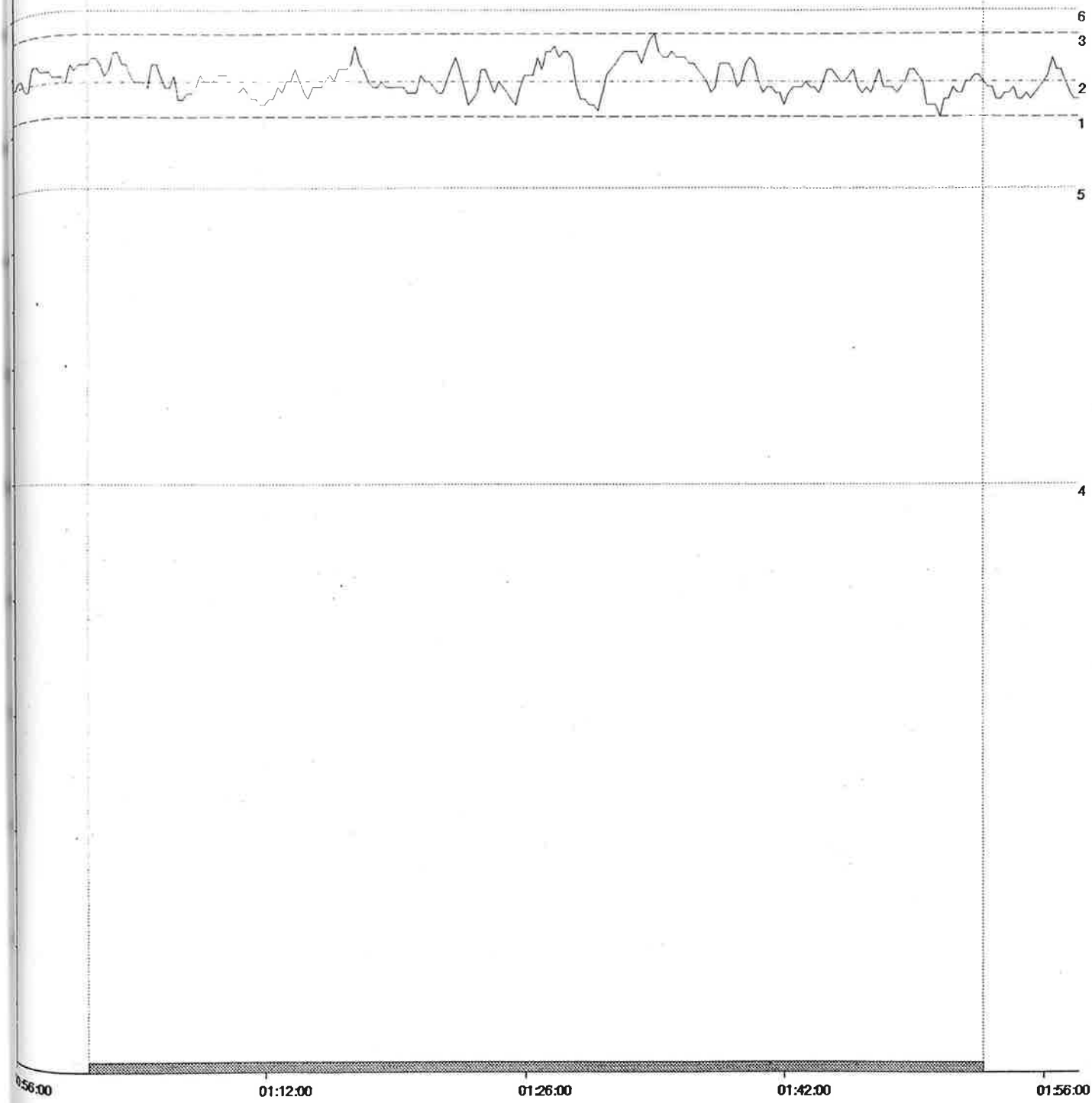
Viikonpäivä	Klo	Leposyke	Verryttely (min)	Harjoitus (min)	Harjoittelumuoto
Ma .../...	...-...	.....	.....	.....	.....
Ti .../...	...-...	.....	.....	.....	.....
Ke .../...	...-...	.....	.....	.....	.....
To .../...	...-...	.....	.....	.....	.....
Pe .../...	...-...	.....	.....	.....	.....
La .../...	...-...	.....	.....	.....	.....
Su .../...	...-...	.....	.....	.....	.....

- Muista alku- ja loppuverryttely sekä venyttely. Verryttelyssä samaa harjoitusmuotoa tehdään kevyemmin eli n. 20 sykettä varsinaista harjoittelua alemmalla syketasolla.
- Merkitse harjoitus 5 minuutin tarkkuudella yllä olevaan tilaan.
- Harjoituspäivänä mittaa leposyke sängyssä makuulla ennen ylösnousua.
- Älä harjoittele, jos tunnet olosi huonoksi, flunssaiseksi jne. Siirrä harjoitus toiseen päivään mikäli vointisi paranee.
- Älä siirrä harjoitusta toiselle viikolle, vaan pyri pitäytymään kunkin viikon ohjelmassa.

LIITE 5

FitWare-ohjelman tärkeimmät tulosteet 3 (8)

Harjoituksen sykekäyrä

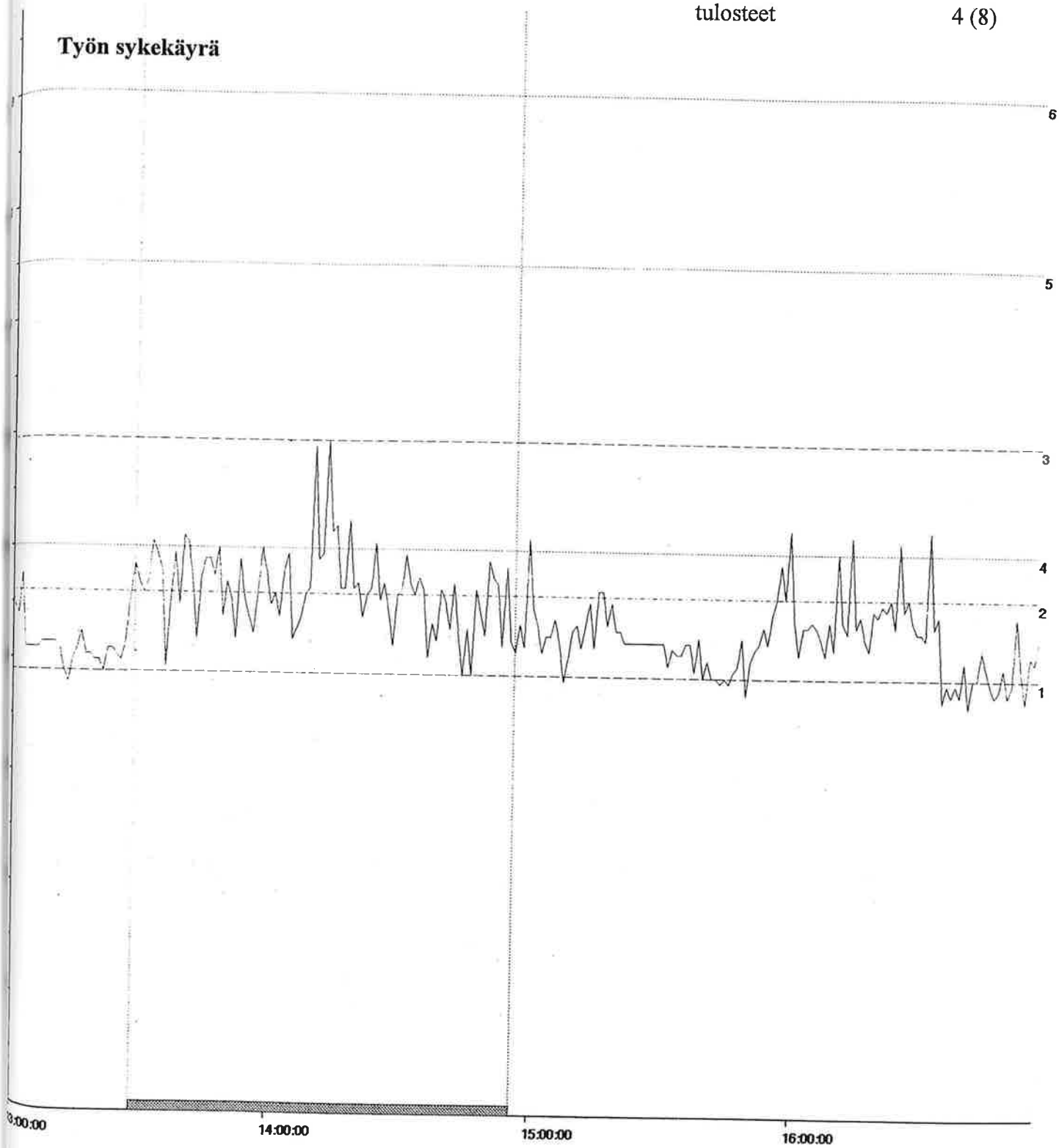


Nimi:	Vellu Partanen (Helsinki Marathon)			Pvm:	20.8.1996	Aika:	11.45
Harjoituksen kesto:	03:00:30	Valittu aikavāli:	01:01:15 - 01:53:00 (00:51:45)				
Anaerobinen kynnys (6):	180	Min. Syke (1):	162	Keskisyke (2):	168	Max. syke (3):	176
Aerobinen kynnys (5):	150	Max. METs:	18,1	Fyysinen kuormittavuus [METs]:	14,8	% Max. METs:tä:	81,8
Peruskestävyyden alaraja (4):	100	Energian kulutus [kJ]:	3878	Energian kulutus [kcal]:	926		

LIITE 5

FitWare-ohjelman tärkeimmät tulosteet 4 (8)

Työn sykekäyrä



Nimi:	Vellu Partanen (Helsinki Marathon)			Pvm:	13.07.1996	Aika:	09.27.39
Työn kesto:	19:21:00	Valittu aikaväli:	13:07:00 - 14:34:00 (01:27:00)				
Anaerobinen kynnys (6):	180	Min. syke (1):	78	Keskisyke (2):	92	Max. syke (3):	119
Aerobinen kynnys (5):	150	Max. METs:	18,1	Fyysinen kuormittavuus [METs]:	5,5	% Max. METs:stä:	30,5
Peruskestävyyden alaraja (4):	100	Energian kulutus [kJ]:	2482	Energian kulutus [kcal]:	593		

Kestävyyden eri alueet

äskiliikunta

- Sykealue 50-60% maksimisykkeestä - vastaava syketaajuus k.o. henkilöllä: 95 - 115
- N. 60 % energiantuotosta rasvoja hapettamalla.
- Energiantuotto lähes täysin aerobista (hapenvaraista).
- Pitkäkestoista, matalatehoista, hieman rasittavaa, 'keskusteluvauhtia'.
- Kehittää rasva-aineenvaihduntaa.
- Energiankulutustaso suhteellisen alhainen, rasvaa palaa n.20-40 g/h, joten merkittävään laihtumiseen vaaditaan suuri määrä liikuntaa, mutta se kohdistuu ennen kaikkea ylimääräisen rasvakudoksen määrään.
- Pieni komplikaatioriski.

eruskestävyys

- K.o. henkilöllä sykealue 51-77 % maksimisykkeestä - vastaava syketaajuus: 100-150
- 30-65 % maksimihapenkulutuksesta.
- Hieman rasittavaa, alimmainen paita kostuu, hengitys kiihtyy hieman, 'keskusteluvauhti'.
- Lihastyöstä vastaavat pääasiassa hitaat lihassolut.
- Energiantuottotapa aerobinen, anaerobisesti 1 % tai vähemmän, maitohapon muodostus anaerobisesti hiilihydraateista ei merkittävästi kasva lepotasosta.
- Lihastyön energianlähteenä pääsääntöisesti rasvat (50-60 % energiantuotosta rasvoja happettamalla).
- Peruskestävyysalueen harjoittelu kehittää ennenkaikkea rasva-aineenvaihdunnan tehokkuutta, huonompikuntoisella myös sydämen toimintatehoa.
- Harjoittelun tulee olla pitkäkestoista (yli 30 min) tällä sykealueella.
- Huomattava osa liikunna terveysvaikutuksista (mm. verenpaineen lasku, osa rasva- ja hiilihydraattiaineenvaihdunnan vaikutuksista) saavutettavissa jo tällä tasolla.
- Pieni sydänkomplikaatioiden ja rasitusvammojen riski.
- Peruskestävyysalueen ylärajoilla nopeiden lihassolujen käyttöönotto ja siten anaerobisten energiantuottomekanismien osuus lisääntyy ja veren maitohappopitoisuus alkaa nousta lepotasosta; saavutetaan aerobinen kynnys.

Kestävyyden eri alueet

Vauhtikestävyys

- K.o. henkilöllä sykealue 77-92 % maksimisykkeestä - vastaava syketaajuus: 150-180
- 65-87 % maksimihapenkulutuksesta.
- Melko rasittavaa, selvä hengästyminen ja hikoilu, puhe sujuu vielä.
- Lihastyöstä vastaavat hitaat ja enenevästi nopeat lihassolut.
- Energiantuottotapa aerobinen, 2-10 % anaerobisesti, maitohapon muodostus merkittävästi suurempaa kuin levossa, elimistö kykynee puskuroimaan maitohaposta johtuvan happamuuden nousun.
- Lihastyön energianlähteenä 50-80 %:sti hiilihydraatit (mitä kovempi teho, sen suurempi osuus).
- Vauhtikestävyys kehittyy 15-60 minuutin harjoitteilla tällä sykealueella.
- Terveysvaikutuksina sekä rasva- että hiilihydraattiaineenvaihdunta paranee, sydän- ja verenkiertoelimistön toiminta tehostuu merkittävästi, maksimihapenkulutus paranee myös.
- Pieni sydänkomplikaatioiden ja hieman suurentunut rasitusvammojen riski.
- Vauhtikestävyysalueen ylärajoilla nopeiden lihassolujen käyttöönotto lisääntyy entisestään, veren maitohappopitoisuus nousee voimakkaasti, elimistö ei enää pysty puskuroimaan happamuuden lisääntymistä; saavutetaan anaerobinen kynnys ja seuraa lihasten väsyminen kuormituksen jatkuessa.

Maksimikestävyys

- K.o. henkilöllä sykealue 92-100 % maksimisykkeestä - vastaava syketaajuus: 180-195
- 87-100 % maksimihapenkulutuksesta.
- Suurin mahdollinen hengitys- ja verenkiertoelimistön toiminnan teho.
- Hyvin rasittavaa, voimakas hengästyminen ja hikoilu.
- Lihastyöhön rekrytoitu sekä nopeat että hitaat lihassolut.
- Energiantuotto maksimikestävyystasolla suorituksen alussa anaerobista, pitkittyessään enemmän aerobista, esim. 10 min kohdalla jo 85 %:sti.
- Energianlähteenä 80-90 %:sti hillihydraatit.
- Tämä sykealue kehittää erityisesti maksimaalista hapenkulutusta ja anaerobista kestävyyttä.
- Harjoitustyyppi kovatehoinen, tasavauhtinen kuormitus 5-30 min tai intervalli-harjoitukset.
- Positiivisia terveysvaikutuksia vähemmän kuin alempitasoisilla kuormituksilla.
- Selvästi suurentunut rasitusvamma- ja sydänperäisten komplikaatioiden riski.

LIITE 5
FitWare-ohjelman tärkeimmät
tulosteet 7 (8)

Ennakkokyselykaavake testiin saapuvalle

mi:_____ Ikä:_____ Paino:_____ Pituus:_____

ko sinulla todettu joku seuraavien sairausryhmien sairauksista ?

Hengitys- ja verenkiertoelinten sairaudet:_____

Aineenvaihduntasairaudet:_____

Tuki- ja liikuntaelinsairaudet:_____

Muut sairaudet:_____

Lääkitys:_____

ko sinulla ollut rintakipuja ?

__ Levossa __ Rasituksessa __ Ei lainkaan

ko lähisuvussa sydänsairauksia ?

__ Isä __ Äiti __ Veli __ Sisko

Mitä:_____

ko hengenahdistuksia ?

__ Levossa __ Rasituksessa

imausta / tajunnanmenetyskohtauksia ?

__ Levossa __ Rasituksessa

ko 2 viikon sisällä ollut ?

__ Kuumetta __ Flunssaa __ Yskää

pakointi ?

__ Ei __ <5 __ 10 __ 15 __ 20 __ 25 __ >30 (Savuketta/vrk)

Lopettanut vuonna:_____ __ Savukkeita __ Sikaria __ Piippua

an työn fyysinen rasittavuus:

__ Toimisto __ Kevyt ruumiillinen __ Raskas ruumiillinen

matkat: _____ km

__ Autolla __ Pyörällä __ Kävellen

LIITE 5
FitWare-ohjelman tärkeimmät
tulosteet 8 (8)

Ennakkokyselykaavake testiin saapuvalle

Liikuntaharrastukset: _____

_____

Kilpaurheilu: _____

_____

Liikunta-aktiivisuus 'hengästyen ja hikoillen' viimeisen 3 kk:n aikana?

__ Ei lainkaan __ Kerran viikossa

__ 2-3 kertaa viikossa __ Säännöllisesti yli 4 kertaa viikossa

Liikunnan kerta-annos:

__ Alle 30 min __ 30-60 min __ yli tunti

Kuntoarviosi samanikäiseen suomalaisväestössä (lihaskunto)?:

__ Heikko __ Välttävä __ Keskitaso __ Hyvä __ Erinomainen

Kuntoarviosi samanikäiseen suomalaisväestössä (kestävyyskunto)?:

__ Heikko __ Välttävä __ Keskitaso __ Hyvä __ Erinomainen

TULE TESTIIN OIKEIN VALMISTUNEENA!

1. Vältä edellisenä päivänä raskasta fyysistä rasitusta ja alkoholin käyttöä.
2. EI TUPAKKAA väh. 2 tuntia ennen testiä
3. EI KAHVIA väh. 2 tuntia ennen testiä.
4. Ateriasta vähintään 2 tuntia.
5. Testissä hikoilet ja hengästyt, joten ota mukaan asianomainen varustus ja peseytymisvälineet.

LIITE 6

Esimerkki FitWare-ohjelman

ohjelmakoodista 1 (5)

```
'*****************************************************************************
'          Ohjelma:   FitWare
'          Versio:    1.2
'          Oikeudet:  Fit-Test Oy
'          Yksikkö:   Hiiren liikkuminen sykekäyrällä
'          Funktiot:  HRGRAPH_MouseDown
'                     HRGRAPH_MouseMove
'                     HRGRAPH_MouseUp
'                     EraseOldPaint
'          Globaalit: TEST_PERSON-olio, joka sisältää testattavan kaikki tiedot
'                     PLACE_MAX  - maalauskohdan koordinaattitietoja
'                     PLACE_MIN  - "
'                     OLD_PLACE  - "
'                     TIME_MAX   - "
'                     TIME_MIN   - "
'                     BACK_CLR   - taustan väri
'                     GRID_CLR   - apuviivan väri
'                     SELECT_CLR           - valitun alueen väri
'                     XSTEP      - kahden sykepisteen välinen pikseliero
'                     CHANGED    - lippu, joka ilmoittaa muutoksista
'
'                     MakeGraph            - sykekäyrän piirto
'                     Results              - tuloksien laskenta ja näyttö
'                     CompareToNorms       - tuloksien vertaus normiin
'                     DrawHRLine           - sykesuuntaviivan piirto
'          Kuvaus:    Yksikkö, jonka avulla hoidetaan hiiren ohjaukset sykekäyrällä
'          Pävitetty: 23.2.1996
'          Tekijä:    MA
'*****************************************************************************

'*********************************************************************
'    Ohjelma:           FitWare
'    Versio:            1.2
'    Funktio:           HRGRAPH_MouseDown
'    Tuloparametrit:    button as Integer  -    Hiiren nappitieto
'                       Shift as Integer   -    Näppäimistötieto
'                       X as Single        -    Hiiren x-koordinaatti
'                       Y as Single        -    Hiiren y-koordinaatti
'    Lähtöparametrit:   -
'    Pävitetty:         23.2.1996
'    Tekijä:            MA
'*********************************************************************

Sub HRGRAPH_MouseDown (button As Integer, Shift As Integer, X As Single, Y As Single)
    On Error Exit Sub                'Virhetilanteessa poistutaan aliohjelmasta
    If TEST_PERSON.ACCESS = 0 Then   'jos sykkeitä ei ole purettu niin
    Exit Sub                         'poistutuaan aliohjelmasta
    End If
    EraseOldPaint                    'Poistetaan vanha maalaus
    TEST_PERSON.R1 = 0                   'Nollataan koordinaatit
    TEST_PERSON.R2 = 0
```

LIITE 6

Esimerkki FitWare-ohjelman

ohjelmakoodista 2 (5)

```
        TIME_MIN = 0
        TIME_MAX = 0
        OLD_PLACE = X          'Tallennetaan uudet koordinaatit
        PLACE_MAX = X
        PLACE_MIN = X
End Sub


'*********************************************************************
'    Ohjelma:           FitWare
'    Versio:            1.2
'    Funktio:           HRGRAPH_MouseMove
'    Tuloparametrit:    button as Integer  -   Hiiren nappitieto
'                       Shift as Integer   -   Näppäimistötieto
'                       X as Single        -   Hiiren x-koordinaatti
'                       Y as Single        -   Hiiren y-koordinaatti
'    Lähtöparametrit:   -
'    Pävitetty:         21.2.1996
'    Tekijä:            MA
'*********************************************************************

Sub HRGRAPH_MouseMove (button As Integer, Shift As Integer, X As Single, Y As Single)
    Dim hght      as integer
    Dim wdth      as integer
    Dim step      as single
    Dim ystep     as integer
    Dim yystep    as single
    Dim total_sec as long
    Dim hour      as integer
    Dim min       as integer
    Dim sec       as integer
    Dim cur_time  as string

    On Error GoTo Virhe              'Virhetilanteessa siirrytään kohtaan Virhe
    If TEST_PERSON.ACCESS = 0 Then   'Jos sykkeitä ei ole purettu niin
        Exit Sub                     'poistutaan aliohjelmasta
    End If

    'Lasketaan ja näytetään hiiren kursoria vastaava aika ja sykkeet
    hght = HRGRAPH.Height - (ERGO.Height / 10)              'hght=sykekäyrän korkeus
    ystep = Fix(hght / ((TEST_PERSON.HR_MAX + 30) / 10))
    yystep = ystep / 10
    If TEST_PERSON.HR_INTERVAL < 60 Then                'Jos syketallennusväli on
        step = Int((X - 286) / XSTEP)                   'pienempi kuin 60 s, niin
    Else                                                'käytetään desimaalimuotoa
        step = ((X - 286) / XSTEP)                      'sekuntien näyttämiseksi
    End If

    total_sec = ((step)) * TEST_PERSON.INTERVAL         'Hiiren kursoria vastaava
    'aika sekunteina
    If total_sec > 0 Then                               'Jos kok.aika > 0
        hour = Int(total_sec / 3600)                    'Muutetaan kokonaisaika
```

LIITE 6

Esimerkki FitWare-ohjelman

ohjelmakoodista 3 (5)

```
        min = Int((total_sec Mod 3600) / 60)          'tunneiksi, minuuteiksi ja
        sec = Int(total_sec Mod 60)                   'sekunneiksi

        cur_time = Format(hour,"00")+":"+Format(min,"00")+":"+Format(sec,"00")
        TimeLabel.Caption = cur_time                  'Muotoiltu aika TimeLabel-
                                                      'komponentille =
                                                      'aika näytölle
        TimeHR.Caption = Format(TEST_PERSON.HR(Int(step)), "000") 'Aikaa vastaava
                                                                  'syke näytölle

        'Jos hiiren kursori sallitulla alueella, niin kursoria
        'vastaava syke näytölle
        If Y <= hght Then
            CurHR.Caption = Format(((hght - Y) / yystep), "000")
        End If

        'Jos hiiren nappia on painettu JA hiiri liikkuu oikealle
        'Maalaa hiiren liikkeen mukaan oikealle
        If button = 1 And X > OLD_PLACE And X > PLACE_MAX Then
            HRGRAPH.Line (OLD_PLACE, hght - 15)-(X, hght - 75), SELECT_CLR,BF
            HRGRAPH.Line (PLACE_MIN, hght - 15)-(OLD_PLACE, hght - 75), BACK_CLR, BF

            PLACE_MIN = X
            PLACE_MAX = OLD_PLACE

        'Jos hiiren nappia painettu JA hiiri liikkuu oikean rajan yli
        'Maalaa oikean reunan rajaan asti
        ElseIf button = 1 And X > OLD_PLACE And X <= PLACE_MAX Then
            HRGRAPH.Line (X, hght - 15)-(PLACE_MAX, hght - 75),BACK_CLR, BF
            HRGRAPH.Line (PLACE_MIN, hght - 15)-(OLD_PLACE, hght - 75),BACK_CLR,BF

            PLACE_MIN = X
            PLACE_MAX = OLD_PLACE

        'Jos hiiren nappia on painettu JA hiiri liikkuu vasemmalle
        'Maalaa hiiren liikkeen mukaan vasemmalle
        ElseIf button = 1 And X < OLD_PLACE And X < place_min Then
            HRGRAPH.Line (X, hght - 15)-(OLD_PLACE, hght - 75), select_clr,BF
            HRGRAPH.Line (OLD_PLACE, hght - 15)-(PLACE_MAX, hght - 75),BACK_CLR, BF
            PLACE_MIN = X
            PLACE_MAX = OLD_PLACE

        'Jos hiiren nappia on painettu JA hiiri liikkuu vasemman rajan yli
        'Maalaa vasemman reunan rajaan asti
        ElseIf button = 1 And X < OLD_PLACE And X > place_min Then
            HRGRAPH.Line (PLACE_MIN, hght - 15)-(X, hght - 75), BACK_CLR, BF
            HRGRAPH.Line (OLD_PLACE, hght - 15)-(PLACE_MAX, hght - 75),BACK_CLR, BF
            PLACE_MIN = X
            PLACE_MAX = OLD_PLACE
        End If
End If
Exit Sub
```

LIITE 6

Esimerkki FitWare-ohjelman

ohjelmakoodista 4 (5)

```
Virhe:
     msgbox "Virhe sykekäyrän analysoinnissa!",48    'Ilmoitetaan virheestä käyttäjälle
     Exit Sub
     Resume Next
End Sub


'*************************************************************************
'    Ohjelma:           FitWare
'    Versio:            1.2
'    Funktio:           HRGRAPH_MouseUp
'    Tuloparametrit:    button as Integer  -    Hiiren nappitieto
'                       Shift as Integer   -    Näppäimistötieto
'                       X as Single        -    Hiiren x-koordinaatti
'                       Y as Single        -    Hiiren y-koordinaatti
'    Lähtöparametrit:   -
'    Pävitetty:         23.2.1996
'    Tekijä:            MA
'*************************************************************************

Sub HRGRAPH_MouseUp (button As Integer, Shift As Integer, X As Single, Y As Single)
     Dim hght       As Integer
     Dim wdth       As Integer
     Dim step       As Integer
     Dim ystep      As Integer
     Dim yystep     As Single

     If TEST_PERSON.ACCESS = 0 Then    'Jos sykkeitä ei ole syötetty niin
          Exit Sub                     'poistutaan aliohjelmasta
     End If

     On Error GoTo Error               'Virhetilanteessa siirrytään kohtaan Error
     screen.MousePointer = 11          'Muutetaan hiiren kursori tiimalasiksi
     wdth = ERGO.HR_GRAPH.Width - (ERGO.HR_GRAPH.Width / 10)  'wdth=sykekäyrän leveys
     HR_GRAPH.Cls                      'Tyhjennetään sykekäyrä
          step = ((PLACE_MIN - 285) / XSTEP)
     If step < 0 Then step = 0         'Askel ei voi olla negatiivinen
     TIME_MIN = Int(step * TEST_PERSON.INTERVAL)     'Maalauksen aloituskohta sekunteina

     step = ((PLACE_MAX - 285) / XSTEP)
     If step > TEST_PERSON.TOTALHR Then step = TEST_PERSON.TOTALHR      'Askel ei voi
                                                                       'olla
                                                                  'koknaissykemäärää
                                                                  'suurempi
     TIME_MAX = Int(step * TEST_PERSON.INTERVAL          'Maalauksen lopetuskohta
     sekunteina
     TEST_PERSON.R1 = TIME_MIN
     TEST_PERSON.R2 = TIME_MAX
     MakeGraph                         'Piirretään sykekäyrä
     Results                           'Lasketaan ja näytetään tulokset
     CompareToNorms NORMS              'Verrataan tuloksia normistoon
     DrawHRLine                        'Piirretään sykesuuntaviiva
     CHANGED = True                    'Testattavan tietoja on muutettu
```

LIITE 6

Esimerkki FitWare-ohjelman

ohjelmakoodista 5 (5)

```
    screen.MousePointer = 0                 'Hiiren kursori takaisin oletuskursoriksi
    Exit Sub
Error:
    MsgBox "Virheellinen valinta!", 48      'Ilmoitetaan virheestä käyttäjälle
    Resume Next
    Exit Sub
End Sub


'************************************************************************
'    Ohjelma:           FitWare
'    Versio:            1.2
'    Funktio:           EraseOldPaint
'    Tuloparametrit:    -
'    Lähtöparametrit:   -
'    Pävitetty:         23.2.1996
'    Tekijä:            MA
'************************************************************************

Sub EraseOldPaint ()
    Dim hght      As Integer
    Dim place_1   As Integer
    Dim place_2   As Integer

    'Määritetään maalauksen aloitus- ja lopetuskohdat
    hght = HRGRAPH.Height - (ERGO.Height / 10)
    place_1 = Int(((TIME_MIN * XSTEP) / TEST_PERSON.INTERVAL) + 285)
    place_2 = Int(((TIME_MAX * XSTEP) / TEST_PERSON.INTERVAL) + 285)
    'Maalataan vanhan maalauksen päälle taustavärillä
    HRGRAPH.Line (place_1, hght - 15)-(place_2, hght - 75), BACK_CLR, BF
End Sub
```

Kuvailulehti

MIKKELIN AMMATTIKORKEAKOULU - MIKKELIN TEKNILLINEN OPPILAITOS
Osasto, linja
Sähköosasto, tietotekniikan suuntautumisvaihtoehto

Tekijä
Mika Ahvenranta

Työn nimi
FITWARE-ANALYYSIOHJELMAN SUUNNITTELU JA OHJELMOINTI FIT-TEST OY:LLE

Työn laji	Aika	Sivumäärä + liitteet
suunnittelu / toteutus	19.5.1997	70 + 18

Työn valvoja	Työn ohjaaja
Markku Nuutinen	Hannu Rahikainen

Tiivistelmä

Työni tarkoituksena oli toteuttaa FitWare-analyysiohjelma Windows-ympäristöön Mikkeliläiselle Fit-Test Oy:lle. FitWare-ohjelma tarkentaa ja tekee helpommaksi ihmisen kestävyyskunnon mittauksen, ja sen avulla voidaan seurata myös työpäivän ja harjoituksen fyysistä kuormittavuutta. Ohjelma käyttää analyyseissään hyväksi polkupyöräergometriä ja Polar Electro Oy:n tietokonepurettavia sykemittareita.

Työn tehtäviin kuuluivat tiedon hankkiminen vaatimusmäärittelyä varten, ohjelman suunnittelu ja ohjelman koodaus. Ohjelmistokehityksen muut vaiheet, kuten testaus ja ylläpito eivät kuuluneet tehtäviin.

Aloitin työn keräämällä käyttäjältä ja käytettäviltä asiantuntijoilta tietoa ohjelmiston vaatimusmäärittelyä varten. Vaatimusmäärittelyn valmistuttua, ohjelma mallinnettiin Microsoftin Visual Basic-kehitintyökalun avulla. Mallinnuksella paikattiin vaatimusmäärittelyn puutteita ja hahmoteltiin lopullisen ohjelman ulkonäkö ja toiminnot. Mallinnusta käytettiin useaan otteeseen, jotta ohjelman ulkonäkö ja toiminnot saatiin hiottua halutuiksi. Valitsin Visual Basicin myös FitWare-ohjelman ohjelmointikieleksi, jonka avulla muutosten tekeminen ohjelmaan oli helppoa ja nopeaa.

Työn tuloksena syntyi valmis tuote Fit-Test Oy:lle. FitWare-ohjelmaa käytetään tällä hetkellä mm. työterveysasemilla, kuntoutuslaitoksissa ja puolustusvoimissa. Työstä kertyi työnantajalle myös arvokasta tietoa ohjelma- kehityksen eri vaiheiden ajankäyttöön ja kustannuksiin liittyvistä asioista FitWare-ohjelman jatkokehittelyä varten. Itse sain insinöörityöstä arvokasta kokemusta ohjelmistokehityksen eri työvaiheista, Windows-ohjelmoinnista ja tiimityöskentelystä.

Avainsanat:
ohjelmistotekniikka, Windows-ohjelmointi

Luottamuksellisuus
Osittain salainen

Description page

MIKKELI POLYTECHNIC - MIKKELI INSTITUTE OF TECHNOLOGY
Department
Electric, Computer science

Author
Mika Ahvenranta

Name
DESIGNING OF FITWARE-ANALYSIS SOFTWARE FOR FIT-TEST LTD

Subject	Time	Pages + appendix
Software design	19.5.1997	70 + 18

Supervisor	Instructor
Markku Nuutinen	Hannu Rahikainen

Abstract

The purpose of this design was to create a Windows based analysis and follow-up programme called FitWare. It can be used for analysing physical exercise capacity by using bicycle ergometer and heart rate monitors and interface devices by Polar Electro Ltd. Effects of exercise and physical occupational stress can also be analysed.

I began the work by picking up the information for the programme by interviewing the employer and specialists available. After the requirement specification was completed, I started software modelling with Microsoft's Visual Basic development tool. Modelling the software was essential at this point, because designing the graphical user interface and functions of FitWare program could be made by modelling. The missing features of the requirement specification were also added by modelling the software. Changes were made all the time during the development of FitWare programme, therefore I chose Visual Basic also as final programming tool for FitWare.

As a result of my work, a complete product, FitWare analysis programme, was created for Fit-Test Ltd. Today FitWare program is used, for example, in occupational health care centers, rehabilitation centers and also in Finnish armed forces. The company also got valuable information on matters concerning timing and financing of software development. This information facilitates the future planning of FitWare programme.

Keywords
software designing, Windows programming

Confidentially
Partly secret