VTT TIEDOTTEITA – MEDDELANDEN – RESEARCH NOTES 1913

Testausympäristön konfigurointityökalun käytettävyyden parantaminen

Mika Salmela
VTT Elektroniikka

VALTION TEKNILLINEN TUTKIMUSKESKUS
ESPOO 1998

ISBN 951–38–5313–6 (nid.)
ISSN 1235–0605

ISBN 951–38-5314–6 (URL: http://www.inf.vtt.fi/pdf/)
ISSN 1455–0865 (URL: http://www.inf.vtt.fi/pdf/)

JULKAISIJA – UTGIVARE – PUBLISHER

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

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

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

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

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

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

Toimitus Kerttu Tirronen

LIBELLA PAINOPALVELU OY, ESPOO 1998

Salmela, Mika. Testausympäristön konfigurointityökalun käytettävyyden parantaminen. Espoo 1998: Valtion teknillinen tutkimuskeskus, VTT Tiedotteita – Meddelanden –Research Notes 1913. 56 s.

Avainsanat software usability, user interfaces, configuration tools

Tiivistelmä

Tietokoneohjelmistojen käytettävyys on noussut merkittäväksi teemaksi ohjelmistoalalla. Ohjelmistojen käytettävyyttä voidaan parantaa lähinnä kehittämällä niiden käyttöliittymiä. VTT Elektroniikassa toteutettiin projekti, jossa testausympäristön konfigurointia pyrittiin parantamaan kehittämällä visuaalisuutta hyväksikäyttävä konfigurointityökalu.

Tutkimuksen tarkoituksena on selvittää, kuinka visuaalisuus ja graafinen käyttöliittymä ovat parantaneet MOSIMin konfigurointieditorin käytettävyyttä. Lisäksi selvitetään, kuinka käytettävyyttä voidaan lisätä työkaluun ja kuinka käytettävyyttä mitataan.

Ohjelmistojen käytettävyyttä voidaan parantaa mm. iteroivalla prototypointiprosessilla. Suunniteltavasta ohjelmistosta tuotetaan prototyyppi, jota arvioidaan sopivalla käytettävyyden arviointimenetelmällä. Saatuja tuloksia voidaan käyttää ohjelmiston uudelleensuunnittelussa.

ConfigToolissa käytettävyyttä lisättiin kehittämällä siihen graafinen käyttöliittymä, jossa käytetään yleisiä suoran käsittelyn menetelmiä, kuten ikkunoita, valintalistoja ja kuvakkeita. ConfigTooliin kehitetyllä käyttöliittymällä pyritään vähentämään konfiguraatioon tulevia virheitä, nopeuttamaan konfiguroinnin oppimista ja tehostamaan työskentelyä. Tehtyjä konfiguraatioita pyritään havainnollistamaan visuaalisilla kuvauksilla.

Käytettävyyden arviointimenetelmiä on kehitetty useita. Tässä tutkimuksessa käytettiin Nielsenin kehittämää heuristista arviointimenetelmää. Lisäksi työkalua arvioitiin Eisenstadin ja kollegoiden kehittämällä arviointikehikolla.

Suoritetut tutkimukset osoittavat, että visuaalinen työkalu ConfigTool helpotti MOSIMin konfiguraation tekemistä. Verrattuna vanhaan manuaaliseen konfigurointimenetelmään ConfigToolin käyttö on nopeampi oppia ja sillä voidaan tuottaa konfiguraatioita nopeammin. Lisäksi työkalun rakenne ja esitystapa helpottivat käyttäjiä hahmottamaan tekemänsä konfiguraatiot paremmin.

Alkusanat

Käytettävyysteema alkoi kiinnostaa jo ensimmäistä harjoitustyötäni tehdessä Oulun yliopistossa. Työn aiheena oli referoida Normanin kirja Miten avata mahdottomia ovia. Siitä lähtien olen pohtinut, millaisilla menetelmillä päästään hyvään käytettävyyteen ja mikä tekee jostakin ohjelmasta käytettävän. Tässä tutkielmassa olen miettinyt näitä teemoja, ja tutkielman tehdessäni olenkin saanut paljon tietoa käytettävyyden parantamisesta.

Työ tehtiin VTT Elektroniikan MOTS 2 -projektissa, ja se on hyväksytty opinnäytteenä Oulun yliopistossa.

Erityisesti tutkielman aloittaminen ja tutkimusongelmien määrittely oli tuskaista. Tässä vaiheessa ja myöhemminkin minua auttoivat ohjaajani fil. toht. Ilkka Tervosen antamat neuvot, joista olen erittäin kiitollinen. Lisäksi kiitän professori Veikko Seppästä ja fil. toht. Kari Kuuttia kommenteista ja neuvoista. Tahtoisin myös kiittää hyvästä yhteistyöstä työtovereitani dipl.ins. Petri Jurmua, ins. Tuomo Kähköstä, fil. maist. Juha Lehtikangasta ja dipl.ins. Tero Mannista.

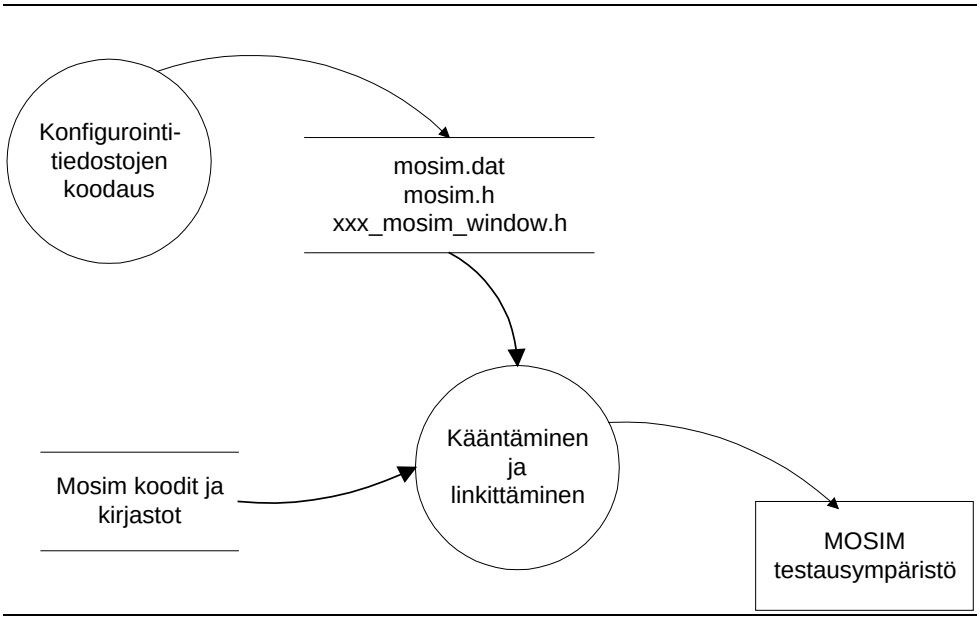
Mika Salmela

Sisällysluettelo

1. Johdanto

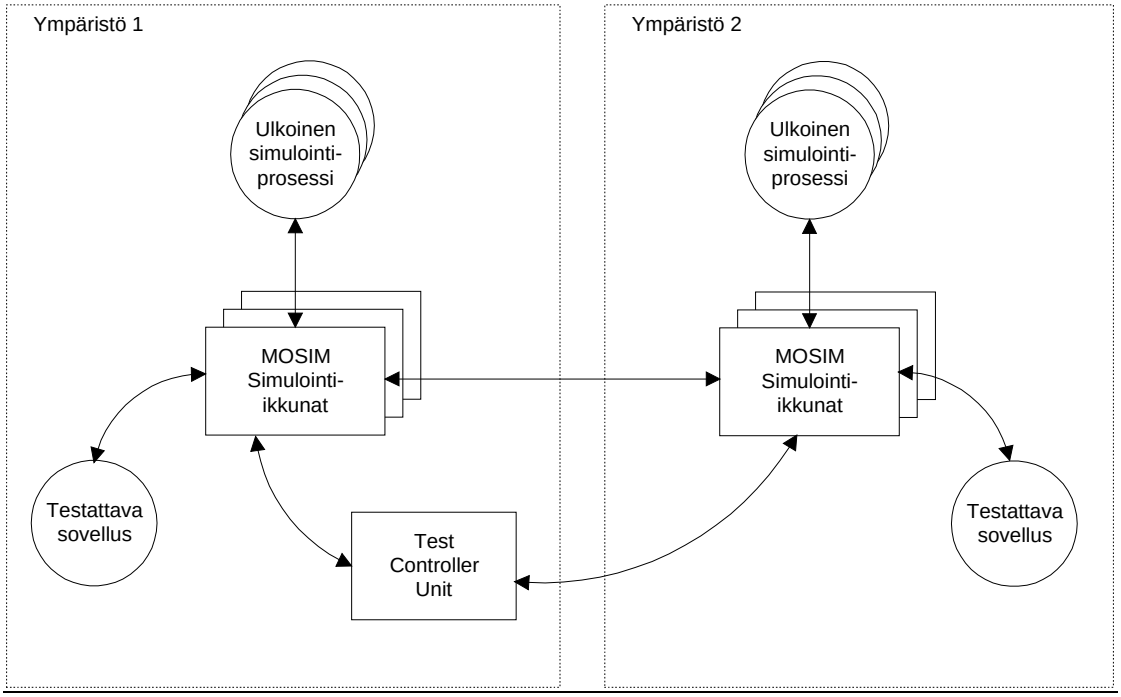
Tutkimuksessa toteutettiin MOSIM ConfigTool -niminen työkalu, jonka ominaisuuksia julkaisussa arvioidaan.

Projektin tarkoituksena oli toteuttaa MOSIM-testausympäristöön visuaalinen konfigurointityökalu, jolla jatkossa määritellään kaikki MOSIMin konfigurointiin tarvittavat tiedot. Perinteisesti MOSIMin konfigurointi on suoritettu tekemällä ASCII-pohjaisia tiedostoja. Kuvassa 1 on esitetty prosessi MOSIM-testausympäristön konfiguroimisesta vanhalla manuaalisella menetelmällä. Prosessi alkaa konfigurointitiedostojen manuaalisella koodaamisella. Kun konfigurointitiedostot ovat valmiita, ne käännetään ja linkitetään yhdessä MOSIMin muiden koodien ja kirjastojen kanssa. Lopputuloksena saadaan ajovalmis testausympäristö.



Kuva 1. MOSIMin konfigurointiprosessi.

MOSIM-ympäristö koostuu komponenteista (kuva 2). Kuvan esimerkissä testausympäristö koostuu kahdesta ympäristöstä, joissa kummassakin on testattava sovellus. Test Controller Unit toimii ohjausyksikkönä, jonka avulla testien suoritusta ohjataan. Kommunikointi testattavaan ohjelmaan ja ulkoisiin simulointiprosesseihin tapahtuu simulointi-ikkunoiden kautta. Simulointi-ikkunoiden syöttökentistä syötetään viestejä ja signaaleja testattavaan ohjelmaan ja ulkoisiin simulointiprosesseihin. (Honka 1992, 70-72).



Kuva 2. MOSIM-ympäristön komponentit ja niiden liittäminen toisiinsa.

MOSIMin konfiguroinnissa määritellään, mistä komponenteista ympäristö koostuu, missä komponentit sijaitsevat ja kuinka ne on liitetty toisiinsa. Lisäksi määritellään kuinka komponentit kommunikoivat keskenään. MOSIMin konfigurointi on tehty perinteisesti suoritettu kirjoittamalla manuaalisesti kolmea eri tyyppiä olevia tiedostoja: mosim.dat, mosim.h ja <window_name>_mosim_window.h.

Mosim.dat -tiedostossa määritellään verkon konfigurointi eli ympäristön sisältämät komponentit, niiden sijainti ja kommunikointiväylät toisiin ympäristöihin. MOSIM-komponentit voivat sijaita fyysisesti eri koneissa, jolloin tässä tiedostossa määritellään verkko-osoitteet komponenteille.

Mosim.h -tiedostossa määritetään, kuinka testattava ohjelma liitetään testaustyökaluun ja kuinka ohjelma ottaa ohjausviestejä vastaan.

Simulointi-ikkunat määritellään tiedostoissa <window_name>_mosim_window.h. Määrittelyssä kuvataan ikkunan ulkoasu, ikkunasta lähtevät viestit ja signaalit sekä simulointifunktiot, joilla voidaan simuloida testattavan sovelluksen todellista ympäristöä.

Perinteisen konfigurointimenetelmän käytettävyydessä havaittiin suuria puutteita. Käyttäjillä tuli paljon virheitä pelkästään väärin kirjoitetuista komennoista ja

viitetunnuksista. Konfigurointimenetelmän oppiminen vaati huomattavan paljon koulutusta. Lisäksi käyttäjien oli vaikeata hahmottaa tekemiänsä konfiguraatioita. Näitä ongelmia pyrittiin ratkaisemaan kehittämällä testausympäristön konfigurointiin työkalu, ConfigTool, jossa olisi graafinen käyttöliittymä ja jossa käytettäisiin visuaalisia kuvauksia konfiguraation esittämiseen. Konfigurointityökalun tavoitteena on vapauttaa käyttäjä konfigurointitiedostojen manuaaliselta koodaamiselta.

1.1 Tutkimusongelma

Tietokoneohjelmien tulisi olla helposti opittavia, oikein käytettyjä, turvallisia ja houkuttelevia, eivätkä ne saisi turhauttaa eivätkä väsyttää käyttäjää. Yhä enemmän ohjelmistojen laatua on mitattu käyttäjätyytyväisyydellä, tiedustelemalla mitä ohjelmistojen käyttäjät ajattelevat mahdollisuuksistaan vaikuttaa ohjelmistoihin. (Lund & Tschirgi, 1991).

Käytettävyys on siis merkittävä teema ohjelmistoalalla. Käytettävyyden parantamisesta ja mittaamisesta on tehty lukuisia tutkimuksia ja julkaisuja. Kuinka käytettävyyttä parannetaan? Human-Computer Interaction traditiossa painotetaan loppukäyttäjän mukaanottamista ohjelmistojen suunnitteluvaiheeseen ja käyttäjien tarpeiden syvällistä ymmärtämistä. Kun ohjelmistoja kehitetään prototypoimalla, voidaan prototyyppiä testata loppukäyttäjien toimesta ja saada näin kommentteja kehitystyöhön.

Mukaillen ISO:n määritelmää käytettävyydestä voidaan sanoa, että ConfigTool on hyvin käytettävä jos käyttäjät kykenevät tuottamaan konfiguraatioita tietyllä määrällä koulutusta, vaivaa ja ajan kulutusta. Konfiguraatioita on myös kyettävä tuottamaan tehokkaasti ja ylläpidettävästi. Koulutuksen, ajan kulutuksen ja tehokkuuden tavoitearvoiksi voidaan asettaa, että konfiguraatioita on kyettävä tuottamaan nopeammin ja vähemmällä koulutuksella kuin perinteisellä menetelmällä. Lisäksi käyttäjien asenteiden konfiguraatioiden tekemiseen on oltava myönteisiä.

Tämän tutkimuksen tarkoituksena on selvittää, millaisilla ratkaisuilla päästään asetettuihin tavoitteisiin ConfigToolin käytettävyydessä sekä kuinka hyvin ratkaisut toimivat. Tutkimusmenetelmänä on konstruktiivinen tutkimus. Tutkimuksen lähtötilanteena on testausympäristön manuaalinen konfigurointi ja lopputilanteena visuaalinen konfigurointityökalu. Tutkielman pääongelma voidaan esittää muodossa:

- Miten konfigurointityökalun käytettävyyttä voidaan parantaa?

Tutkielman mielenkiintoisia lisäkysymyksiä ja alaongelmia ovat:

- Miten konfigurointityökalun käytettävyys määritellään?
- Mitä visuaalisuus tarkoittaa konfigurointityökalun tapauksessa?
- Miten visuaalisuuden lisääminen konfigurointityökaluun tapahtuu?
- Paraniko MOSIM-konfigurointityökalun käytettävyys ConfigToolin avulla?

Tutkielmaan tarvittava teoreettinen tausta graafisista käyttöliittymistä, visuaalisuudesta, prototypoinnista ja käytettävyyden arvioinnista hankittiin kirjallisuustutkimuksella. Käytettävyyttä arvioitiin usealla eri menetelmällä. Käyttäjien asenteita ja henkilökohtaisia mielipiteitä ConfigToolista kerättiin kyselylomakkeella. Käytettävyysongelmien kartoitus tehtiin Nielsenin kehittämällä heuristisella tutkimusmenetelmällä (Nielsen 1994). Konfigurointityökalun toimivuutta visuaalisena työkaluna arvioitiin käyttämällä Eisenstadin ja kumppaneiden kehittämää arviointikehikkoa (Eisenstad et al. 1990).

Luvussa 2 esitellään, millaisilla menetelmillä käytettävyyttä voidaan parantaa ja miksi niitä on käytetty Configtoolin kehittämiseen. Luvun loppuosassa esitetään prosessi, kuinka käyttöliittymiä tulisi kehittää, jotta niiden käytettävyyttä voitaisiin tehostaa. Luvussa 3 esitetään ne käytettävyyden arviointimenetelmät, joita on käytetty ConfigToolin arvioinnissa. Luvussa 4 on esitetty yksityiskohtaisemmin testausympäristön konfigurointityökalu ConfigTool. Luvussa 5 tarkastellaan kuinka käytettävyys ConfigToolin avulla on muuttunut verrattuna manuaaliseen menetelmään. Luvussa 6 arvioidaan ConfigToolin toimivuutta ja käytettävyyttä esitettyjen arviointimenetelmien avulla.

2. Menetelmiä käytettävyyden parantamiseen

MOSIMin perinteisessä konfigurointimenetelmässä oli käytettävyyden kannalta muutamia vakavia puutteita. Käyttäjillä tuli paljon virheitä pelkästään väärin kirjoitetuista komennoista ja viitetunnuksista. Konfigurointimenetelmän oppiminen vaati huomattavan paljon koulutusta. Lisäksi käyttäjien oli vaikeata hahmottaa tekemiänsä konfiguraatioita. Niiden tekeminen oli hidasta ja työlästä.

Näitä puutteita pyrittiin korjaamaan kehittämällä visuaalisuutta tukeva työkalu konfigurointien tekemiseen. Työkalun graafisella käyttöliittymällä pyritään vähentämään virheiden esiintymistä ja nopeuttamaan oppimista. Lisäämällä työkaluun visuaalisia kuvauksia pyrittiin auttamaan käyttäjiä hahmottamaan paremmin tekemänsä konfiguraation.

Tässä kappaleessa käsitellään menetelmiä, joilla käytettävyyttä pyrittiin parantamaan MOSIMin konfigurointityökalussa.

2.1 Visuaaliset ohjelmointityökalut

Ohjelmoitaessa perinteisillä ohjelmointikielillä, tulee lukuisia virheitä pelkästään virheellisesti kirjoitetuista käskyistä ja muuttujien nimistä. Tämän vuoksi on kehitetty visuaalisia ohjelmointityökaluja, joissa on vähennetty tarvetta kirjoittaa tekstiä manuaalisesti. (Bragg, 1996).

Grafiikan käyttö kuvaamaan lukuja, käsitteitä ja teknisiä esityksiä on yleistynyt viime aikoina. Kuviot voivat esittää paljon enemmän kuin tekstuaaliset esitykset. Parhaimmillaan kuvioita voidaan käyttää välineinä esittämään ideoita ja loogista informaatiota. (Chang 1990, 89).

Shu N. on esittänyt seuraavanlaiset lähtökohdat visuaalisuuden käytölle (Shu 1998, 7).

- Kommunikoinnissa kuvat ovat tehokkaampia kuin sanat. Kuvat voivat välittää enemmän sisältöä suppeammassa ilmaisuyksikössä.

- Kuvat auttavat ymmärtämisessä ja muistamisessa.

- Kuvat voivat tarjota kiihokkeita ohjelmoinnin harjoittelulle.

- Kuvilla ei ole kielimuureja. Riippumatta siitä, mitä kieltä ihminen puhuu, hän voi ymmärtää kuvia, jotka ovat oikein suunniteltuja.

Niin sanotun Deutsch-rajoituksen mukaan visuaalisessa ohjelmoinnissa näytöllä voi olla korkeintaan noin 50 perusalkiota. Luku 50 ei ole ehdoton raja, mutta on kumminkin selvää, että mitä enemmän näytöllä on alkioita, sitä sekavammaksi näytettävä asia muuttuu. (Wozniewich 1996).

Vaikka kuvat ovat tehokkaita havainnollistamaan, on joitakin asioita joita ei visuaalisesti pystytä esittämään. Esimerkiksi on vaikeata monien abstraktien käsitteiden ilmaiseminen visuaalisesti, kuten luokan virtuaalisen metodin. (Wozniewich 1996).

Visuaalisessa ohjelmoinnissa myös abstrakteille käsitteille voidaan luoda jokin symboli. Jos näitä käsitteitä on paljon, nousee erilaisten symbolien lukumäärä suureksi. Jos kaikille symboleita ei ole voitu kehittää hyvin kuvaavaa esitysmuotoa, symbolien tulkinnassa tulee vaikeuksia. Ohjelmoijat joutuvat muistamaan kunkin symbolin merkityksen. Symbolien merkityksen opetteluun kuluu paljon aikaa. Vastaavaa olisi jos länsimainen henkilö joutuisi opettelemaan kiinan kielen merkistön.

Tietokoneen ja käyttäjän välinen kommunikaatio voidaan toteuttaa monella eri tavalla. MOSIMin konfigurointieditorissa, ConfigToolissa, on käytetty suoraa käsittelyä. Suoraa käsittelyä on toteutettu tekemällä ConfigTooliin graafinen käyttöliittymä, jossa on käytetty mm. ikkunointitekniikkaa, valintaikkunoita, valikoita ja kuvakkeita.

2.2 Suora käsittely

Suorassa käsittelyssä käyttäjä vaikuttaa siihen, mitä näytöllä tapahtuu. Tällä tavalla käyttäjä tuntee ohjaavansa fyysisesti objekteja näytöllä. Suoran käsittelyn etuja ovat (Booth 1989, 51):

- Aloittelijat oppivat nopeasti perustoiminnot, yleensä kokeneemman käyttäjän opastuksella.

- Kokeneemmat käyttäjät selviytyvät nopeasti erilaisista tehtävistä, kehittäen samalla uusia toimintoja ja ominaisuuksia.

- Satunnaisen käyttäjät pystyvät muistamaan ohjelmiston käyttötavan.

- Virheilmoituksia tarvitaan harvoin.

- Käyttäjät havaitsevat heti ovatko he pääsemässä tavoitteeseensa. Jollei suunta ole oikea, toimintoja voidaan muuttaa nopeasti.

- Käyttäjien ahdistuneisuus laskee, koska järjestelmä on helppotajuisempi ja toiminnot voidaan perua.

Suoraa käsittelyä toteuttaviin järjestelmiin kuuluu yleensä (Preece 1993, 82):

- ikkunoita, jotka jakavat näytön eri alueisiin,
- kuvakkeita, jotka edustavat olioita ja joita voidaan liikuttaa näytöllä,
- hiiri (tai jokin muu osoitinlaite), jolla näytön olioita voi käsitellä sekä
- ponnahdus- tai alasvetovalikoita, jotka näyttävät mitä valintamahdollisuuksia on.

Vaikkakin suora käsittely on hyödyllinen tapa toteuttaa tietokoneen ja käyttäjän välinen vuorovaikutus, siinäkin on ongelmia. Graafiset esitysmuodot, jotka ovat tavallisia suorassa käsittelyssä, eivät sovellu kaikkiin tapauksiin. Kuitenkin suorassa käsittelyssä käyttäjä hallitsee vuorovaikutusta. Tämä antaa käyttäjälle tunteen luotettavuudesta. (Booth 1989, 51).

Suoraa käsittelyä toteuttavia järjestelmiä on alettu kutsua myös graafisiksi käyttöliittymiksi, jotka sisältävät myös paljon grafiikkaa. On havaittu, että hyvin suunnitellut käyttöliittymät ovat aloittelijoille helppokäyttöisiä. Graafisten käyttöliittymien etuja ovat mm. (Preece 1993, 82):

- Koko järjestelmä on helposti esitettävissä, koska objektit esitetään kuvakkeina ja käytettävissä olevia valikko-komentoja voidaan selailla.
- Perustoiminnot, kuten avaaminen, sulkeminen, kopiointi, tuhoaminen ja vieritys, ovat yhdenmukaisia koko järjestelmässä. Tämä helpottaa järjestelmän oppimista.
- Järjestelmän tutkiminen on helppoa, koska suoritetut toiminnot voidaan perua jollei niillä ole toivottua vaikutusta.

2.3 Valintaikkunat

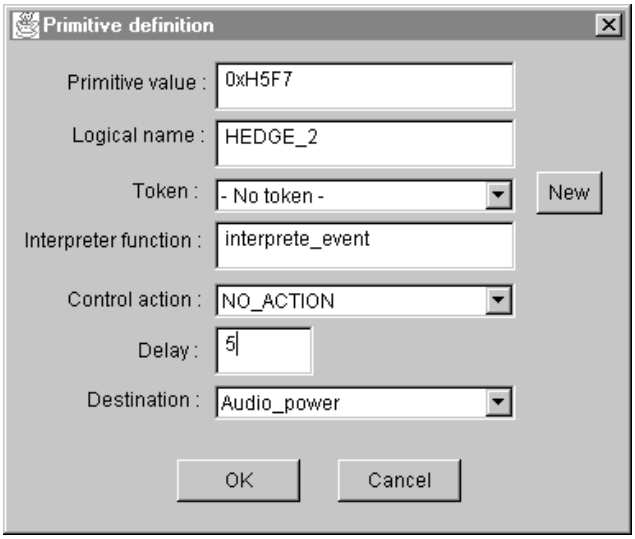
Kun käyttäjän täytyy syöttää näppäimistöllä useaa eri tyyppiä olevaa tietoa, on hyödyllistä tehdä näyttö lomakemuotoon. Tämä on soveliasta varsinkin kun jotakin tietoa joutuu syöttämään useasti. Eräs tapa tehdä lomakkeet helpoksi käyttää on suunnitella ne muistuttamaan vastaavia paperilomakkeita. (Preece 1993, 80).

Lomakemuotoiset dialogit antavat käyttäjälle vähän vaikutusmahdollisuuksia. Käyttäjä voi täyttää vain näkyvissä olevat kentät ja käyttää painikkeita, joita dialogiin sisältyy. Etuna on, että käyttäjän ei tarvitse muistaa komentoja tai niiden syntakseja. (Booth 1989, 49).

Virheiden kokonaismäärä vähenee huomattavasti, jos järjestelmä pystyy ehkäisemään kirjoitusvirheiden syntymisen (Huang 1990, 79). Käyttämällä valintalistoja saadaan käyttäjä valitsemaan juuri oikeassa muodossa olevat arvot, eikä kirjoitusvirheitä pääse syntymään. Lisäksi lomakepohjaisilla dialogeilla käyttäjälle saadaan viestitettyä, mitä kenttiä hänen tulee täyttää ja mitä arvoja hänen tulee käyttää.

Lomakepohjaisia dialogeja käyttämällä voidaan vähentää virheiden määrää. Mikään ei kumminkaan takaa, että valinnat ovat semanttisesti oikeita.

Kuvassa 3 on esimerkki ConfigToolissa käytettävästä lomakepohjaisesta valintaikkunasta. Kolmeen kenttään (*Token*, *Interpreter function* ja *Destination*) käyttäjä voi valita arvot vain valintalistoista. Muihin kenttiin arvot syötetään näppäimistöllä.



Kuva 3. Esimerkki ConfigToolissa käytettävästä lomakepohjaisesta dialogista primitiivin määrittelyyn.

2.4 Kuvakkeet

Jos halutaan esittää graafisesti käyttäjän luomia komponentteja ja antaa käyttäjälle mahdollisuus käsitellä niitä, joudutaan käyttämään hyväksi kuvakkeita. Kuvakkeiden avulla voidaan toteuttaa suoran käytön periaatetta. Näytöllä oleviin kuvakkeisiin käyttäjä voi kohdistaa erilaisia toimenpiteitä käyttäen esimerkiksi hiirtä.

Kuvakkeet ovat pieniä graafisia kuvia, joita käytetään ilmentämään erilaisia piirteitä käyttöliittymän kielikuvista. Niihin kuuluvat järjestelmän oliot, valintamahdollisuudet, toiminnot, sovellukset ja viestit. Tavallisesti kuvakkeita käytetään siten, että käyttäjä osoittaa jotain kuvaketta ja valitsee sen. Jatkotoimintona saattaa olla kuvakkeen liikuttaminen tai aukaiseminen. (Preece 1993, 75).

Kuvakkeet eivät ole oikeastaan kuvia, vaan abstrakteja ja tyyliteltyjä hahmoja. Kuvakkeet auttavat tuttujen symbolien välittömässä tunnistamisessa, mutta oudossa ympäristössä kirjaimet, numerot ja sanat ovat tehokkaampia. Kuvakkeet voidaan jakaa kuva- ja symbolikuvakkeisiin. Kuvalliset kuvakkeet käyttävät kuvia esittämään abstraktia ja semanttista tietoa operaatioista. Kuvakkeita, jotka käyttävät hahmoja esittämään semanttista informaatiota, kutsutaan symbolikuvakkeiksi. (Huang 1990, 68).

Tiedonvälitys kuvakkeiden avulla tarkoittaa kuvan käyttöä välittämään ideaa tai toimintaa ei-sanallisella tavalla. Kuvat tai hahmot voidaan valita kuvakkeisiin yhdennäköisyyden tai samankaltaisuuden perusteella. Kuviot voidaan valita myös aikaisemmin määritellyistä ja opituista kuvista ja merkeistä. Jotta kuvakkeet tulkittaisiin oikein, niissä täytyy olla oikea kuva, otsikko ja asiayhteys. (Chang 1990, 3).

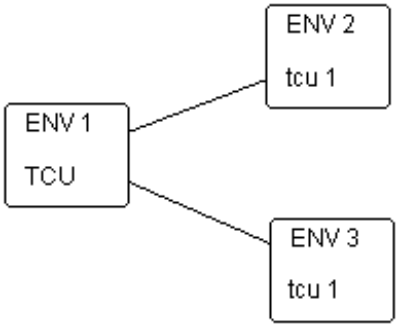
Kuvakkeiden käytön etuna verrattuna perinteisiin komentoihin on, että useissa tapauksissa ne on helpompi oppia ja muistaa. Tämä johtuu siitä, että: (Preece 1993, 76).

- kuvakkeet tarjoavat enemmän visuaalista informaatiota käsiteltävästä oliosta.
- kuvakkeet tarjoavat helposti muistettavan vihjeen ja
- kuvakkeet esittävät selkeästi järjestelmän olioiden väliset suhteet.

Kuvakkeiden käytössä on myös ongelmia ja haittoja. Jotkut kuvakkeet ovat luonnostaan moniselitteisiä ja joskus kuvakkeet voidaan tulkita vain tietyssä asiayhteydessä. Koska ei ole joukkoa yleisesti hyväksyttyjä kuvakkeita, ne muuttuvat ajan kuluessa. Tämän vuoksi kuvakkeiden suunnittelu on tehtävä huolellisesti. Kuvakkeiden suunnittelu tulisi tehdä kolmessa vaiheessa, (1) valitaan esitysmuoto, (2) tehdään malli ja (3) testataan tuotettu kuvake. (Chang 1990, 4).

MOSIMin konfigurointieditorissa käytetään etupäässä symbolikuvakkeita ilmaisemaan tehtyä määritystä, lisättyä komponenttia. Kuvakkeissa käytetään sanoja ilmaisemaan mitä komponenttia kuvake edustaa. Käyttäjä voi asettaa kuvakkeen valituksi, jolloin komponentille voidaan tehdä muutoksia. Sen parametreja voidaan muuttaa tai komponentti voidaan poistaa konfiguraatiosta. Kuvakkeiden paikkaa voidaan myös muuttaa. Kuvakkeiden asemalla voidaan esittää havainnollisemmin komponenttien suhdetta toisiinsa.

Kuvassa 4 on esitetty esimerkki MOSIM:n konfigurointieditorissa käytettävistä kuvakkeista ja siitä, kuinka niillä informoidaan konfiguraatiosta. Esimerkissä on muodostettu testausympäristö, joka koostuu kolmesta aliympäristöstä. Esimerkissä vasemmanpuoleisin ympäristö ohjaa kahta muuta ympäristöä.



Kuva 4. Esimerkki ConfigToolin kuvakkeista.

Luotaessa tai editoitaessa objekteja näytöllä, on hyödyllistä antaa dynaamista graafista palautetta käyttäjälle. Yksinkertaiset toiminnot, kuten muuttaminen, skaalaus, pyöritys, poistaminen, jne. tosiaankin tuottavat tietoa, mistä antaa palautetta. Raahaaminen on tärkeä keino esittämään toiminnon dynaamista muuttumista. Raahattaessa valittua objektia liikutetaan yhtäjaksoisesti paikasta toiseen. (Huang 1990, 75).

Kun ConfigToolin käyttäjä tekee muutoksia komponentteihin, tehdään vastaava muutos välittömästi myös komponenttia esittävään kuvakkeeseen. Käyttäjällä on myös mahdollisuus raahata kuvakkeita näytöllä.

Toimintojen kohdistaminen objekteihin voidaan toteuttaa kahdella eri periaatteella. Verbi-objekti -menetelmässä ensin valitaan toiminta ja sitten toiminnan kohde. Tehokkaampi menetelmä on subjekti-verbi, jossa ensin valitaan kohde ja sitten toiminta. Jälkimmäisessä menetelmässä voidaan esimerkiksi dynaamisesti muuttaa valikoita valitun objektin mukaan. (Huang 1990, 75).

ConfigToolissa käytetään etupäässä subjekti-verbi -menetelmää. Esimerkiksi komponenttien poistaminen ja muuttaminen tapahtuu niin, että ensin valitaan komponentti ja sitten toiminto valikoista.

2.5 Näyttöjen yhtenäistäminen

On tärkeätä suunnitella näytöt niin, että näyttö mahdollistaa käyttäjiä huomaamaan mistä tarvittava tieto löytyy. Tämä saavutetaan parhaiten käyttämällä yhdenmukaisia näyttöjä koko sovelluksessa. (Preece 1993, 72).

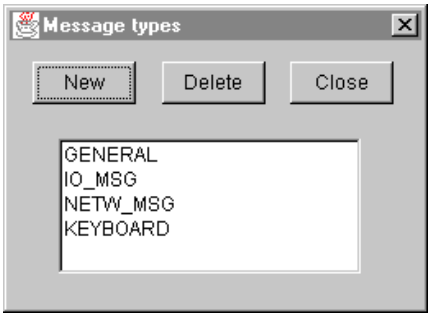
- Tärkeä tieto, joka vaatii välitöntä huomiota, tulisi aina näyttää silmiinpistävällä paikalla niin että käyttäjä huomaa sen.

- Tarpeeton tieto tulisi näyttää vain, jos se helpottaa käyttäjää käsittelemään informaatiota sillä hetkellä.

- Raportit ja viitetieto tulisi ryhmitellä ja esittää näytön reuna-alueilla.

ConfigToolissa yhdenmukaisuutta on pyritty toteuttamaan eri editoreissa ja valintaikkunoissa. Editoreissa työkalupaletti eli toimintopainikkeet on sijoitettu aina samaan paikkaan, oikeaan reunaan. Editorien piirtoalue on sijoitettu keskelle ja valikot yläosaan. Editoreissa tarvittavat samat toiminnot ovat sijoitettu samannimiseen valikkoon. Editorien ikkunoiden otsikkopalkissa näytetään konfiguraation tai komponentin nimi, jota kyseisellä editorilla käsitellään.

Joissakin ConfigToolin toiminnoissa käyttäjä voi luoda uuden komponentin tai muuttaa tai poistaa vanhan komponentin. Kaikki tällaiset toiminnot käsitellään samanlaisissa valintaikkunoissa. Kuvan 5 valintaikkunan *New*-painikkeella voidaan aloittaa uuden komponentin määrittely. *Delete*-painikkeella valittu komponentti voidaan poistaa. Valintaikkunan alaosassa on lista aiemmin määritellyistä komponenteista. Komponentteja voidaan muuttaa kaksoisnäpäyttämällä komponentin nimeä listassa. Toimintaikkunasta aukeava valintaikkuna, jossa komponentti määritellään, sijoitetaan aina toimintaikkunan oikealle puolelle.



Kuva 5. Toimintaikkuna

Valintaikkunat, joihin käyttäjän täytyy syöttää tietoa, sijoitetaan aina keskelle näyttöä. Kun jokin valintaikkuna on aktiivisena, ei muita ikkunoita pysty käsittelemään. Tämä tehdään myös sen vuoksi, että tiedon yhtenäisyys pystytään säilyttämään.

2.6 Työkalun käytettävyyden parantaminen

Käytettävyyden lisääminen tarkoittaa sitä, että järjestelmä täytyy suunnitella intuitiiviseksi ja vaivattomaksi käyttää. Ei ole tarkoituksenmukaista, että järjestelmän

kehitys tapahtuu suunnittelijoille ja järjestelmälle soveliaasti, mutta käyttäjät joutuvat mukauttamaan oman toimintansa järjestelmän mukaan. (Redmond-Pyle & Moore 1995, 6).

Pyrittäessä hyvään käytettävyyteen loppukäyttäjien näkökulmasta, suunnitteluprosessissa tulee toteuttaa seuraavat kolme asiaa: (Redmond-Pyle & Moore 1995, 6).

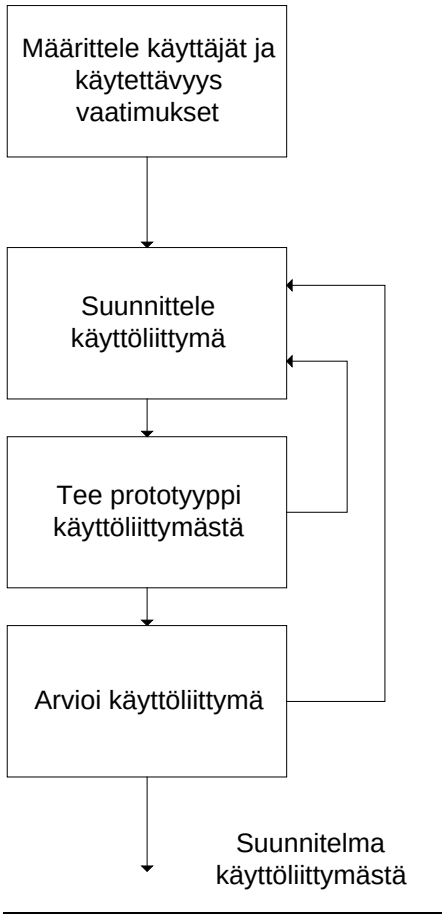
- Suunnittelijoiden täytyy ymmärtää yksityiskohtaisesti, keitä loppukäyttäjät ovat, mitä tehtäviä he suorittavat ja mitkä heidän vaatimuksensa käytettävyydelle ovat.

- Loppukäyttäjien tulee olla aktiivisesti mukana suunnitteluryhmässä analyysistä suunnitteluun. Ei ole riittävää, että käyttäjiltä kysytään heidän vaatimuksensa ja sitten pyydetään heitä arvioimaan suunnitelma.

- Suunnittelijat ja loppukäyttäjät arvioivat yhdessä suunnitelman käytettävyyttä niin aikaisin kuin on mahdollista ja muuttavat suunnitelmaa saatujen tulosten perusteella.

Käytettävyydeltään hyvän järjestelmän suunnittelu on liian vaikea prosessi, jotta se onnistuisi yhdellä suunnittelukierroksella. Vastauksena ongelmaan on esitetty prototypointisilmukkaa, jossa toteutetun prototyypin käytettävyys arvioidaan ja saatuja tuloksia käytetään hyväksi uudelleensuunnittelussa. Kaavio prosessista on esitetty kuvassa 6. (Redmond-Pyle & Moore 1995, 9).

Redmond-Pyle ja Moore ovat kehittäneet laajemman mallin käyttöliittymän suunnittelulle kuin kuvassa 6 on esitetty. ConfigToolia kehittäessä pyrittiin kuitenkin kuvan kaltaiseen prosessiin, joskaan käytännössä suunnittelu ei kuitenkaan onnistunut ideaalisesti. Siksi kannattaa tarkastella, kuinka perusprosessin tulisi edetä.



Kuva 6. Käyttöliittymän suunnitteluprosessi.

Prosessi aloitetaan määrittelemällä, keitä järjestelmän loppukäyttäjät ovat ja asettamalla vaatimukset käytettävyydelle. Tulevasta käyttöliittymästä tehdään ensimmäinen suunnitelma ja sen pohjalta valmistetaan prototyyppi. Prototyypille suoritetaan käytettävyysarviointi, jonka tulosten perusteella suunnitelmaa voidaan tarkentaa.

Käytettävyysarvioinnin tarkoituksena on selvittää, kattaako käyttöliittymä kaikki vaatimukset käytettävyydestä ja tukeeko se riittävästi kaikkia tehtäviä. Avainkysymyksiä ovat: (Redmond-Pyle & Moore 1995, 26).

- Kuinka käytettävä käyttöliittymä on loppukäyttäjien käytössä määriteltyjen vaatimusten näkökulmasta?

- Millaisia ongelmia käyttäjillä ilmeni?

- Tukeeko käyttöliittymä kaikkia mahdollisia tehtäviä, joita käyttäjät tekevät?

3. Konfigurointityökalun käytettävyyden arviointi

Käyttöliittymän tehtävänä on tukea tietokonejärjestelmän käyttöä. Toisin sanoen käyttöliittymän tärkein laatutekijä on sen käytettävyys. Käytettävyyteen sisältyy, että järjestelmä on helppo oppia ja tehokas käyttää. (Redmond-Pyle & Moore 1995, 2).

Booth on esittänyt joukon mittareita, joilla ohjelmiston käytettävyyttä voidaan mitata. Osa mittareista tuottaa määrällistä eli numerotietoa järjestelmän käytettävyydestä. Määrällistä informaatiota tuottavien mittareiden antamia lukuja voidaan käyttää myös tilastollisissa analyyseissä. Osa mittareista on taas luonteeltaan laadullisia ja näiden mittarien antamat tulokset vaativat tulkintaa. (Booth 1989, 120).

Aika. Yksi yleisimmin käytetyistä käytettävyyden mittareista on tehtävän suorittamiseen käytetty aika. Ajan mittaamisen etuja on, että aikaa on helppo mitata ja sitä voidaan käyttää tilastollisiin analyyseihin. Ongelmana on, ettei aina ole selvää, mihin saatuja tuloksia pitäisi verrata. Usein verrataan aloittelijan käyttämää aikaa ammattilaisen käyttämään aikaan. (Booth 1989, 120). ConfigToolin tapauksessa voidaan vertailla vanhaa ja uutta menetelmää. Vertailu voidaan tehdä siten, että määritetään jokin osakokonaisuus konfiguroinnissa, joka sitten suoritetaan kummallakin menetelmällä ja vertaillaan kulutettua aikaa. Ongelmana on, kuinka määritellä osatehtävät niin, että ne ovat tehtävämäärältään samat kummallakin tavalla suoritettuna.

Virheet. Käyttöliittymässä tapahtuvat virheet ovat yksi käyttökelpoisimmista informaation lähteistä. Mittaukset, kuten aika, tarjoavat vain summittaisia merkkejä siitä missä käyttäjät kohtaavat ongelmia. Virheet voivat kuitenkin osoittaa missä kohtaa ongelmat sijaitsevat. Lisäksi virheitä tutkimalla voidaan löytää syyt, jotka aiheuttavat ongelmia. (Booth 1989, 121).

Virheistä saadaan käytettävyyden arvioimiseen sekä määrällistä että laadullista informaatiota. Laadulliseen arviointiin virheet tarjoavat erittäin paljon informaatiota. Ilmenevät virheet viittaavat usein kohtiin, joissa ei ole ymmärretty oikein käyttäjien tarpeita tai käsitystä tehtävästä. (Booth 1989, 121).

Suullinen pöytäkirja. Suulliset pöytäkirjat ovat kirjoitettuja lausuntoja käyttäjiltä joko käytön aikana tai käytön jälkeen. (Booth 1989, 122).

Visuaaliset pöytäkirjat. Visuaaliset pöytäkirjat voidaan nauhoittaa esimerkiksi videokameralla. Tällaiset mittaukset ovat luonteeltaan laadullisia ja vaativat tulkintaa. Visuaalisten pöytäkirjojen avulla suunnittelijat voivat osoittaa, missä käyttäjät kohtaavat ongelmia. (Booth 1989, 122).

Visuaaliset skannauskuviot. Silmän liikkeen tutkimukset voivat osoittaa, minne käyttäjän katse kohdistuu ja kuinka pitkäksi aikaa. Tällaiset tutkimukset antavat tietoa mitä käyttäjä katsoo, mihin aikaan ja kuinka pitkään. (Booth 1989, 122).

Järjestelmän käyttömallit. Tutkimalla käyttömalleja voidaan päätellä mitä järjestelmän osia käytetään eniten. Tällaiset tutkimukset saattavat osoittaa, että jotkin järjestelmän osat ovat helpompia käyttää tai jotkin tehtävät ovat helpommin muunnettavissa ohjelmistolla tuettaviksi. (Booth 1989, 123).

Asennemittaukset. Käyttäjien asenteita mitataan yleensä kyselykaavakkeilla ja haastatteluilla. Kysymykset saattavat koskea esimerkiksi käyttäjien mielipidettä järjestelmän opittavuudesta, helppokäyttöisyydestä ja soveltuvuudesta tehtävään. Toinen kiinnostava kysymys on, tunsiko käyttäjä hallitsevansa tehtävää ja tilannetta vai tunsiko hän itsensä turhautuneeksi. Käyttäjien asenteen mittauksista voidaan päätellä, käyttävätkö ja arvostavatko käyttäjät järjestelmää. (Booth 1989, 124).

Joitakin Boothin mainitsemista mittareista voitiin käyttää myös MOSIMin konfigurointieditorin käytettävyyden arvioimiseen. Aikaa voi mitata siitä, kuinka kauan käyttäjillä menee jonkin osatehtävän suorittamiseen. Tehtävä voidaan suorittaa sekä vanhalla menetelmällä että ConfigToolin avulla. Myös käyttöliittymissä tapahtuvia virheitä voidaan kirjata. Ohjelman käyttötapoja ja käyttäjien asennetta voidaan tiedustella kyselykaavakkeiden avulla.

Käytettävyyden arviointiin ja mittaamiseen on kehitetty useita menetelmiä. Tässä luvussa tarkastellaan niitä menetelmiä, joita käytettiin ConfigToolin arviointiin. Mielipidetutkimuksella selvitettiin, millainen työkalu on loppukäyttäjien näkökulmasta. Nielsenin heuristisella tutkimuksella etsittiin yksittäisiä käytettävyysongelmia työkalusta. Eisenstadtin kriteereillä arvioitiin, kuinka hyvin työkalu toimii visuaalisena konfiguroijana.

3.1 Arviointimenetelmistä

Arvioinnissa on pyrkimyksenä kerätä informaatiota järjestelmän käytettävyydestä tai potentiaalisesta käytettävyydestä, jotta sen käyttöliittymää ja tukimateriaalia voitaisiin kehittää tai arvioida (Preece 1993, 108). Suurin hyöty käytettävyyden arvioinnista saadaan, jos arvioinnin tuloksia käytetään käyttöliittymän kehittämiseen suunnittelu- ja toteutusvaiheen aikana. ConfigToolin tapauksessa arviointi tehtiin vasta valmiiseen käyttöliittymään. Saatujen tulosten perusteella konfigurointityökalun käyttöliittymässä on puutteita, joihin jatkokehityksessä tulisi puuttua.

Arviointimenetelmä on prosessi, jolla kerätään tietoa käyttäjän ja tietokoneen välisen liittymän toiminnasta ja käyttäjien asenteista. Arviointimenetelmät voidaan luokitella viiteen ryhmään. (Preece 1993, 109).

- *Analyyttisessa arvioinnissa* käyttäjien suoritusta arvioidaan ennakolta, käyttämällä formaaleja tai puoliformaaleja kuvauksia käyttöliittymästä.

- *Asiantuntija-arvioinnissa* kokeneet käyttöliittymien suunnittelijat arvioivat käyttöliittymää.

- *Havaintoarvioinnissa* havainnoidaan tai tarkkaillaan käyttäjiä kun he käyttävät ohjelmistoa.

- *Mielipidetutkimuksessa* selvitetään käyttäjien subjektiiviset näkemykset käyttöliittymästä.

- *Kokeellisessa arvioinnissa* käytetään tieteellisiä kokeellisia menetelmiä testaamaan oletuksia ohjelmiston käytöstä.

Taulukko 1. Shackelin näkökannat käytettävyyden arvioimiseen.

TEHOKKUUS	Kuinka tehokkaasti tietty määrä käyttäjiä kykenee suorittamaan tietyt tehtävät käyttöliittymän avulla määrätyssä ympäristössä? Tätä tekijää voidaan myös kutsua tuottavuudeksi, koska arvioidaan, kuinka nopeasti käyttäjä suoriutuu tehtävästä. Mitattavia asioita voivat olla mm. Nopeus ja tehdyt virheet.
OPITTAVUUS	Kuinka paljon koulutusta ja harjoittelua vaaditaan, ennen kuin järjestelmän käyttäminen on tehokasta? Jos järjestelmää käytetään satunnaisesti, kuinka paljon kuluu aikaa uudelleen oppimiseen?
MUKAUTUVUUS	Missä määrin käyttöliittymä on yhä tehokas, jos suoritettavia tehtäviä tai käyttöympäristöä muutetaan.
SUHTAUTUMINEN	Kokevatko käyttäjät järjestelmän väsyttäväksi ja turhauttavaksi, vai onko sen käyttö palkitsevaa ja tyydyttävää.

Taulukossa 1 on esitetty Shackelin neljä näkökantaa siihen, mitä käytettävyyden arviointimenetelmillä pitäisi käytettävyydestä arvioida (Booth 1989, 110). Shackelin

näkökohdissa on huomattava, että hän sitoo käytettävyyden arvioinnin parametrit aina käyttäjiin, suoritettaviin tehtäviin, käyttötilanteeseen ja ympäristöön. Kun selvitetään jonkin järjestelmän käytettävyyttä, on selvitettävä ketä ovat käyttäjät ja millaisten tehtävien suorittamisessa käytettävyyttä mitataan. (Redmond-Pyle & Moore 1995, 3).

3.2 Mielipidetutkimus

ConfigToolin käytettävyyttä arvioitiin myös mielipidetutkimuksella. Mielipidetutkimuksen tarkoituksena on selvittää käyttäjien subjektiiviset mielipiteet ohjelmistosta haastattelujen tai kyselylomakkeiden avulla. (Preece 1993, 115).

Haastattelujen tekeminen vaatii huolellista suunnittelua, jotta esitetyt kysymykset koskisivat arvioitavia kohtia käyttöliittymässä. Yleensä suunnitelma tehdään ennen haastattelua, joko valitsemalla käsiteltävät aiheet tai muodostamalla tarkistuslista aiheista ja kysymyksistä. (Preece 1993, 115).

Itse haastattelun kulku on joko rakenteellista tai joustavaa. *Rakenteellisessa haastattelussa* käydään läpi joukko ennalta suunniteltuja kysymyksiä eikä henkilökohtaisia mielipiteitä tarkastella syvemmin. *Joustavassa haastattelussa* on valmiina aiheet, joita käsitellään, mutta järjestystä ei ole asetettu. Haastattelu etenee haastateltavan vastausten ja ajatusten mukaan. (Preece 1993, 115).

Rakenteellinen haastattelu on helpompi suorittaa ja analysoida, mutta tärkeät yksityiskohdat saattavat jäädä huomaamatta. (Preece 1993, 115).

Kyselylomakkeiden tekoon on yleisesti ottaen kaksi vaihtoehtoa. *Avoimissa kysymyksissä* vastaaja voi vapaasti muotoilla vastauksensa. *Suljetuissa kysymyksissä* vastaajan täytyy valita vastauksensa annetuista vaihtoehdoista. (Preece 1993, 115).

Avoimet kysymykset tarjoavat runsaasti tietoa, mutta tuloksia voi olla vaikea analysoida. Suljetut kysymykset on helpompi analysoida, koska niihin on liitetty arviointiasteikko. Arviointiasteikkoa voidaan käyttää esimerkiksi siten, että kysytään, kuinka hyvin ohjelma havainnollistaa jonkin asian, ja sitten vastaaja valitsee vaihtoehdon väliltä “täysin samaa mieltä” ja “täysin eri mieltä”. (Preece 1993, 115).

3.3 Nielsenin heuristinen arviointi

Monet tekniikat voisivat parantaa huomattavasti käytettävyyttä, jos niitä käytettäisiin ohjelmiston kehityksen aikana. Käytettävyystekniikoita ei kuitenkaan juuri käytetä, koska suunnittelijat pitävät niitä liian kalliina, vaikeina ja aikaa vievinä. Tämän vuoksi

on kehitetty ns. *halpa käytettävyystekniikka* (Discount Usability Engineering), jonka menetelmät ovat halpoja, nopeita ja helppoja käyttää. Heuristinen arviointi on yksi käytetyimmistä ns. halvan tekniikan arviointimenetelmistä. (Nielsen 1994, 25).

Heuristinen arviointi on menetelmä, jolla pyritään löytämään käyttöliittymien suunnitelmissa esiintyviä käytettävyysongelmia. Tietoa havaituista ongelmista voidaan käyttää hyväksi iteratiivisessa suunnitteluprosessissa. Arviointi suoritetaan siten, että arvioija käy muutaman kerran käyttöliittymän läpi ja tutkii eri elementtejä ikkunassa ja vertaa niitä listaan tunnettuja sääntöjä käytettävyydestä. Tämä lista on joukko yleisiä sääntöjä, jotka kuvaavat käytettävyydeltään hyvien käyttöliittymien yleisiä ominaisuuksia. Lisäksi arvioija voi vapaasti käyttää muita täydentäviä sääntöjä käytettävyydestä tai tuloksia, jotka saattavat olla relevantteja. (Nielsen 1994, 29). Alla on esitetty Nielsenin esittämä lista säännöistä, joiden avulla käyttöliittymää voidaan arvioida (Nielsen 1994, 30).

1) Järjestelmän tilan ilmeisyys. Järjestelmän tulisi aina tiedottaa käyttäjää siitä mitä järjestelmä on tekemässä antamalla sopivaa palautetta kohtuullisen ajan kuluessa.

2) Järjestelmän ja todellisen maailman välinen yhteys. Järjestelmän tulisi puhua käyttäjän kielellä, niillä sanoilla, fraaseilla ja käsitteillä, jotka ovat tuttuja käyttäjälle. Järjestelmän tulisi seurata todellisen maailman käytäntöjä ja esittää tieto luonnollisessa ja loogisessa järjestyksessä.

3) Käyttäjän hallinta ja vapaus. Koska käyttäjät valitsevat usein toimintoja vahingossa, järjestelmän tulisi tarjota selkeästi osoitettuja poistumisteitä ei-toivotusta tilasta. Järjestelmän tulisi lisäksi tarjota peruutus- ja uudelleentekotoiminnot.

4) Yhdenmukaisuus ja standardimaisuus. Käyttäjän ei pitäisi joutua miettimään tarkoittavatko erilaiset sanat, tilanteet tai toiminnot samaa asiaa.

5) Virheiden ehkäisy. Virheilmoituksia parempi on suunnitella järjestelmä niin, ettei virheitä pääse edes syntymään.

6) Tunnistaminen mieluummin kuin muistaminen. Järjestelmän tulisi tehdä objektit, toiminnot ja vaihtoehdot näkyviksi. Käyttäjän ei tulisi joutua muistamaan mitä informaatiota ikkunassa oli siirryttäessä toiseen ikkunaan. Ohjeet järjestelmän käytöstä tulisi olla näkyvissä tai helposti saatavissa.

7) Käytön joustavuus ja tehokkuus. Järjestelmän tulisi tarjota oikopolkuja kokeneille käyttäjille, niin että järjestelmä kykenee palvelemaan sekä kokemattomia että kokeneita käyttäjiä. Käyttäjien tulisi pystyä räätälöimään usein käytettyjä toimintoja.

8) Esteettinen ja minimalistinen malli. Ikkunoiden ei tulisi sisältää informaatiota, joka on epäolennaista tai harvoin tarvittua. Jokainen ylimääräinen yksikkö tietoa kilpailee tarpeellisten tietoyksikköjen kanssa ja heikentävät niiden suhteellista näkyvyyttä.

9) Käyttäjien avustaminen virheiden tunnistamisessa, määrittämisessä ja virheistä selviämisessä. Virheilmoitukset tulisi ilmaista selväkielisesti, ei käyttämällä koodeja. Virheilmoitusten tulisi selkeästi osoittaa ongelma ja rakentavasti ehdottaa ratkaisua.

10) Opastus ja dokumentointi. Vaikka on parempi, jos järjestelmää pystyy käyttämään ilman dokumentointia, saattaa olla tarpeellista tarjota opastusta ja dokumentaatiota. Tällaisen informaation tulisi olla helposti löydettävissä ja tarpeeksi suppea, kohdistettu käyttäjän tehtävään ja listata konkreettisia toimenpiteitä.

3.4 Visuaalisten ohjelmointiympäristöjen ominaisuuksia

Visuaalisten ohjelmointiympäristöjen ongelmana on löytää hyvä graafinen esitysmuoto ohjelman käyttäytymiselle niin, että se tukee suunnittelijoiden omaa tapaa ratkaista ongelmia. Tämä on yksi tärkeimmistä, mutta vaikeimmin saavutettavista tekijöistä visuaalisissa ohjelmointityökaluissa. Vaikeutta lisää seikka, että kuvaavuus ei ole ainoa huomioon otettava tekijä. Muiden huomioon ottosaattaa johtaa kompleksisiin vaihtokauppoihin ja rajoituksiin ominaisuuksista. (Eisenstadt et al. 1990). Eisenstadt kumppaneineen on luetellut yhdeksän ominaisuutta, jotka hyvä visuaalinen työkalu täyttää ja joiden avulla työkalua voidaan arvioida (taul. 2).

Termi *visuaalinen ohjelmointi* tarkoittaa mielekkäiden graafisten esitysmuotojen käyttämistä ohjelmointiprosessissa. (Shu 1998, 9). MOSIMin konfigurointi ei varsinaisesti ole ohjelmointia. Konfiguroinnista puuttuvat monet ohjelmointiin liittyvät rakenteet. MOSIMin konfigurointityökalulta voidaan vaatia samoja ominaisuuksia kuin Eisenstadt *et al.* edellyttää visuaalisiltaohjelmointityökaluilta

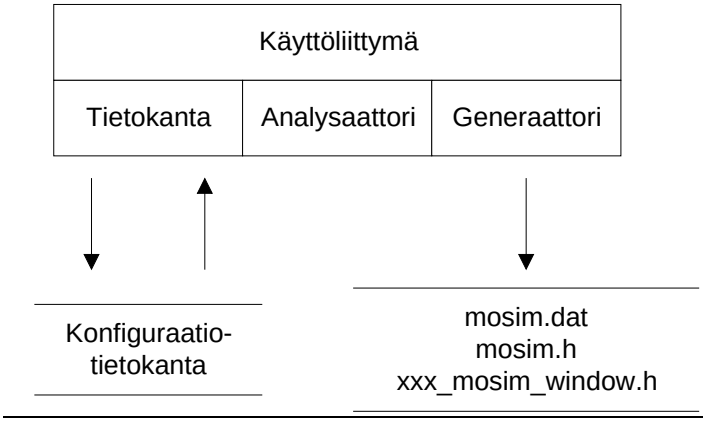
Koska MOSIMin konfigurointi ei ole varsinaisesti ohjelmointia, eivät kaikki Eisenstadin kriteerit sovi täysin konfigurointityökalun arviointiin. ConfigToolilla kattavuus täytettyy, kun sillä pystytään tekemään tietyt tehtävät. Ei voi ennakkoon määrittää kuinka laajoja tai kompleksisia tehtäviä ohjelmointikielellä ratkaistaan.

Taulukko 2. Eisenstadin ja kumppaneiden esittämät kriteerit visuaalisten työkalujen arvioinnille.

1) *Kuvaavuus*	Miten hyvä vastaavuus graafisilla esityksillä on siihen, kuinka ohjelmoija on hahmottanut suunniteltavan asian? Kuvaavuuteen sisältyvät myös selkeän suoritusmallin olemassaolo, tarkoituksenmukaiset visuaaliset esitykset ohjaus- ja tietovirroista sekä kuinka hyvin työkalun alla toimiva virtuaalikone on esitetty.
2) *Käsiteltävyys*	Kuinka hyvin visuaalisesti esityksiä voidaan muuttaa tai päivittää suoraan käyttäjän toimesta?
3) *Tulkittavuus*	Käytetyllä notaatiolla ja siihen liitetyllä toiminnalla täytyy olla itsestään selvä yhteys. Käyttäjän tulee kyetä välittömästi ymmärtämään graafiseen kuvaukseen liitetty toiminto.
4) *Kattavuus*	Kuinka hyvin työkalulla pystytään visuaalisesti ratkaisemaan laajojakin tehtäviä järkevästi ja tuottavasti? Onko työkalun esitystapa sovelias laajojen ongelmien ratkaisemiseen?
5) *Näkyvyys*	Kaikki olemassa olevat komponentit eivät välttämättä ole esillä näytössä oikealla hetkellä, vaikka käyttäjä haluaisi niiden olevan. Näyttääkö työkalu tarvittavat asiat oikealla hetkellä ja voidaanko tarvittavia ominaisuuksia korostaa?
6) *Kytkentä*	Jos jotakin osiota muutetaan, täytyy muutoksen vaikutus ulottua myös niihin osioihin, joilla on jokin yhteys muutettuun osioon. Tätä ominaisuutta kutsutaan sisäiseksi yhdenmukaisuudeksi.
7) *Navigoitavuus*	Hyvän visuaalisen työkalun tulisi helpottaa liikkumista eri osioiden välillä. Hyvä navigoitavuus parantaa myös työkalun kattavuutta (k. 4). Navigointia voidaan parantaa muuttamalla resoluutiota, mittakaavaa, pakkausta, valintatarkkuutta tai abstraktiota (k. 1).
8) Täydellisyys	Kuinka tarkasti työkalu näyttää kaikki suoritetut toimenpiteet? Kuinka tarkasti esitetään kaavioita käsittelevän virtuaalikoneen kytkentä?
9) Käännettävyys	Voidaanko kaikki työkalulla tuotetut kaaviot kääntää koodiksi ja takaisin työkalun esittämiksi kaavioiksi.

4. MOSIMin konfigurointityökalu

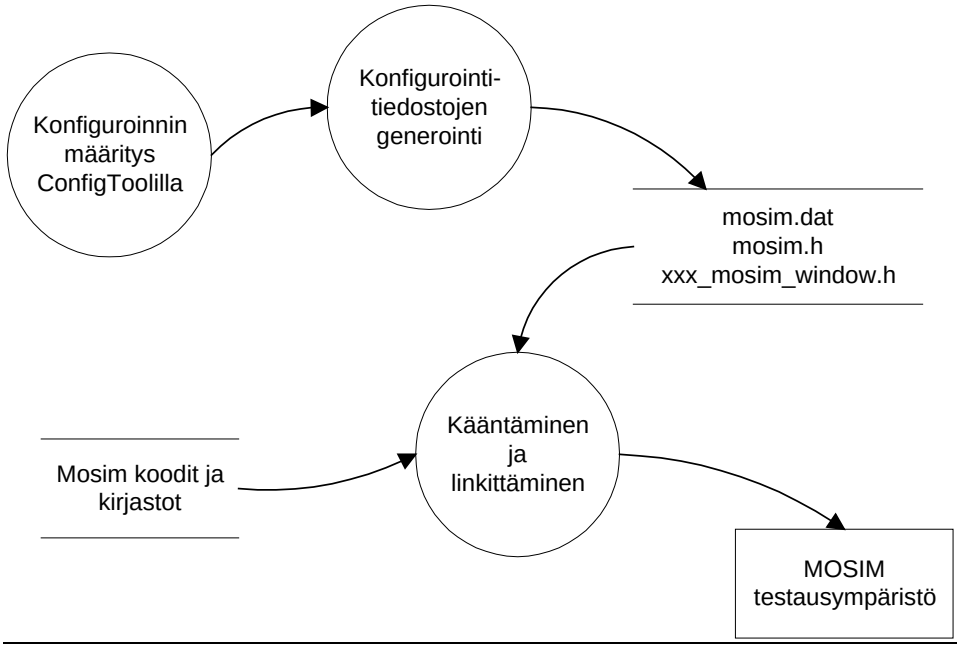
Kuvassa 7 on esitetty ConfigTool-konfigurointityökalun rakenne. Käyttöliittymällä rakennetaan tarvittava testausjärjestelmän ja ohjataan työkalun muita toimintoja. Tietokanta huolehtii käyttäjän tekemien määritysten talletuksesta ja ylläpidosta. Analysaattorilla tarkistetaan tehtyjen määritysten oikeellisuus. Generaattori muodostaa tehdyistä määrityksistä MOSIMin konfigurointiin tarvittavat tiedostot.



Kuva 7. Konfigurointityökalun rakenne.

4.1 Testausympäristön rakentamisprosessi

Kuvassa 8 on esitetty konfigurointiprosessin kulku uudella menetelmällä käyttäen konfigurointityökalua. Aluksi ConfigToolilla muodostetaan graafinen esitys konfiguraatiosta. Tehdyt määritykset tallennetaan tiedostoon, josta muodostetaan Generaattoria käyttäen MOSIM:in konfigurointitiedostot. Lopuksi toimenpiteet ovat samat kuin perinteisessä menetelmässä. Tiedostot käännetään ja linkitetään, jolloin saadaan ajovalmis testausympäristö.



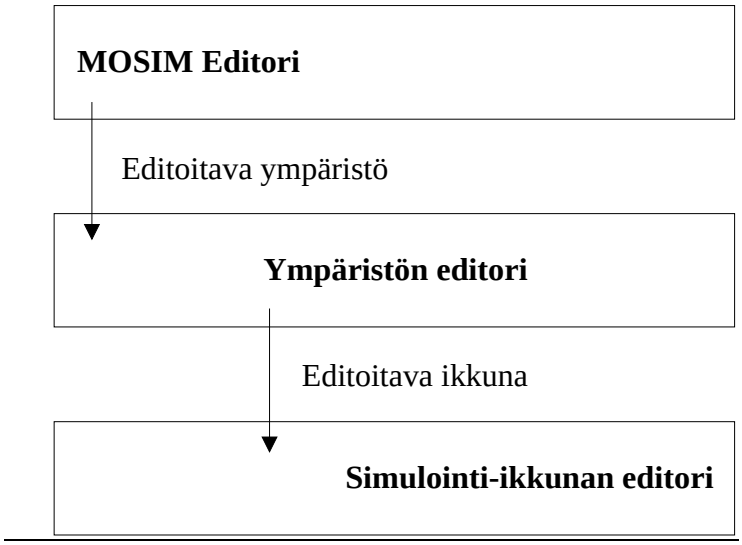
Kuva 8. Konfigurointiprosessi.

4.2 Konfigurointieditori

MOSIMin konfigurointieditori koostuu kolmesta eri työkalusta, kuten on esitetty kuvassa 9. Ylimmällä tasolla (MOSIM Editor) määritellään ympäristöt, niiden sijainti, mikä ympäristö ohjaa mitäkin ja eri ympäristöjen väliset kommunikointikanavat.

Toisella tasolla (ympäristön editori) määritellään ympäristön sisällä olevat komponentit. Mitä ovat simulointi-ikkunat, ulkoiset simulointiprosessit, test controller unit ja testattava sovellus. Lisäksi tällä tasolla määritetään kuinka simulointi-ikkunoista tulevat sanomat välitetään testattavan sovelluksen keskeytyskäsittelijöille.

Kolmannella tasolla (Simulointi-ikkunan editori) määritellään simulointi-ikkunoiden rakenne ja ulkoasu. Tällä tasolla määritellään myös sanomien välittäminen simulointifunktioille.



Kuva 9. Editorin tasot.

MOSIM Editor -tasolla käyttäjä voi siirtyä määrittelemään ympäristön sisälle tulevia komponentteja. Siirtyminen ympäristön editointitasolle tapahtuu kaksoisnäpäyttämällä ympäristön kuvaketta MOSIM Editor -tasolla. Vastaavasti ympäristön määrittely -tasolta käyttäjä voi siirtyä määrittelemään simulointi-ikkunaa kaksoisnäpäyttämällä ikkunan kuvaketta. Työkalussa voi olla yhtä aikaa auki useampia ympäristön ja simulointi-ikkunan määrittelytasoja.

Jokainen työkalutaso koostuu kolmesta osiosta; valikosta, painiketaulusta ja piirtoalueesta. Jokainen valikko sisältää vähintään *Edit*-valikon, jossa on kaksi toimintoa; *Delete* ja *Properties*. Edit-valikon toiminnot kohdistuvat aktiivisena olevaan komponenttiin piirtoalueella. *Delete*-toiminnolla poistetaan komponentti ja *Properties*-toiminnolla muutetaan komponentin parametreja.

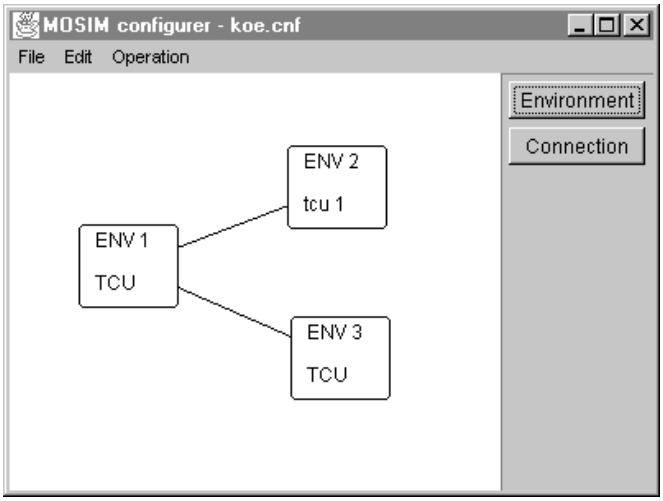
Piirtoalueella näytetään komponentit, jotka käyttäjä on lisännyt konfigurointiinsa. Kuvakkeet ovat liikuteltavissa piirtoalueella, jotta käyttäjä voisi sijoittaa ne kuvaamaan hänen käsitystään konfiguraatiosta.

Painiketaulu sijaitsee työkalun oikeassa laidassa. Taulusta käyttäjä voi lisätä komponentteja konfiguraatioonsa.

4.3 MOSIM Editor

Kuvassa 10 on esitetty konfigurointityökalun päätaso. Tältä tasolta voidaan suorittaa normaalit tallennus-, avaus- ja lopetustoiminnot. Operation-valikosta voidaan käynnistää analysointi ja konfigurointitiedostojen generointi. Tasolta voidaan käynnistää myös editori, jolla kirjoitetaan versiohistoriatietoja.

Ikkunan oikealla puolella on kaksipainikkeinen työkaluvalikko. Environment-painikkeella käyttäjä voi lisätä ympäristöjä konfiguratioonsa. Connection-painikkeella lisätään kommunikointiväylä kahden ympäristön välille.



Kuva 10. MOSIM Editor, työkalun päätaso.

Käyttäjä voi vapaasti liikutella kuvakkeita editorin sisällä sellaiseen muotoon, että se kuvaa käyttäjän mielikuvaa testausympäristöstä paremmin. Ympäristön ominaisuudet määritellään yleensä samalla kun kuvake luodaan. Ominaisuuksia voi kuitenkin muuttaa jälkeenpäin. Käyttäjä valitsee muutettavan ympäristön kuvakkeen ja muuttaa *Edit*-valikon *Properties*-toiminnolla tarvittavia parametreja. Valittu kuvake ilmaistaan tummentamalla sen tausta, kuten taulukossa 3 on esitetty.

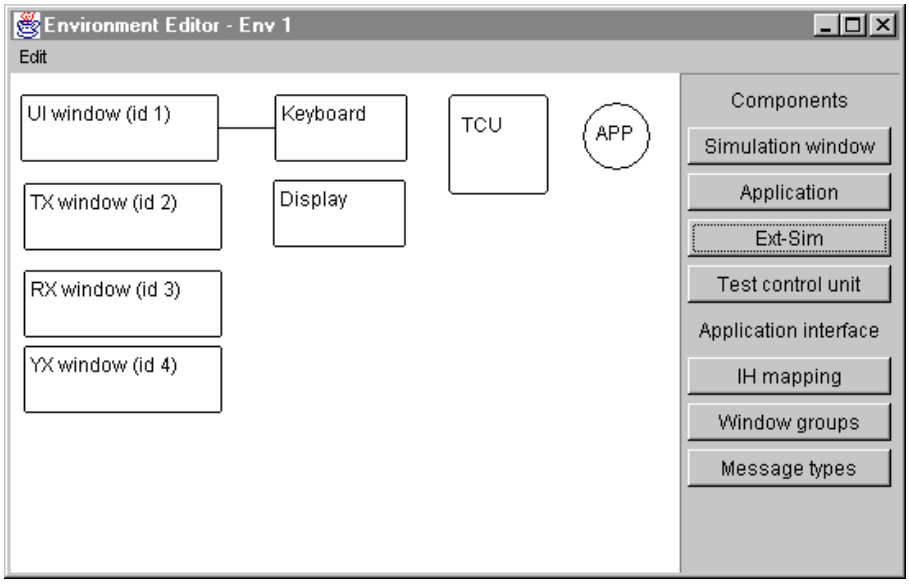
Taulukossa 3 on lueteltu kuvakkeet, joita editorissa käytetään. Ympäristöt esitetään ikkunassa neliönmuotoisena kuvakkeena. Kuvakkeen ylälaidassa näytetään ympäristön tunnistenumero. Mikäli ympäristö sisältää Test Control Unitin, eli on ohjaava ympäristö, on kuvakkeen alalaidassa isoilla kirjaimilla teksti ‘TCU’. Jos joku toinen ympäristö kontrolloi ympäristöä, on kuvakkeen alalaidassa ilmoitettu ohjaavan ympäristön tunnistenumero. Kommunikointikanava ympäristöjen välillä ilmaistaan viivalla kuvakkeiden välillä.

Taulukko 3. MOSIM editorin kuvakkeet.

ENV 1 TCU	Ympäristökuvake. Kuvakkeessa näytetään ympäristön tunnistenumero tekstin ‘ENV’ jälkeen, jotta ympäristöt olisivat tunnistettavissa. Teksti ‘TCU’ ilmaisee, että ympäristö sisältää Test Control Unitin.
ENV 2 tcu 1	Esimerkki toisesta ympäristöstä. Tämän ympäristön ohjausyksikkö sijaitsee ympäristössä nro 1.
ENV 1 TCU	Valittu ympäristökuvake.

4.4 Ympäristöeditori

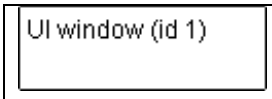
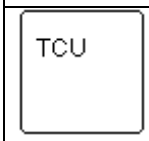
Kuvassa 11 on esitetty ympäristöeditori. Tällä tasolla käyttäjä määrittelee kuinka monta simulointi-ikkunaa ympäristö sisältää, kuinka monta ulkoista simulointiprosessia on käytössä, sekä sen sisältääkö ympäristö Test Controller Unitin ja testattavan sovelluksen. Nämä komponentit esitetään kuvakkeina editorin piirtoalueella. Lisäksi tällä tasolla määritetään, kuinka ikkunoista tulevat viestit ja signaalit ohjataan testattavan sovelluksen keskeytyskäsittelijöille. Ohjaukseen käytettäviä ikkunaryhmiä ja viestityyppejä voidaan myös käsitellä ympäristöeditorista.



Kuva 11. Ympäristöeditori.

Taulukossa 4 on esitetty kuvakkeet joita käytetään ympäristön komponenttien esittämiseen. Simulointi-ikkunasta näytetään sen nimi ja tunnistenumero. Simulointi-prosessista näytetään käyttäjän antama looginen nimi.

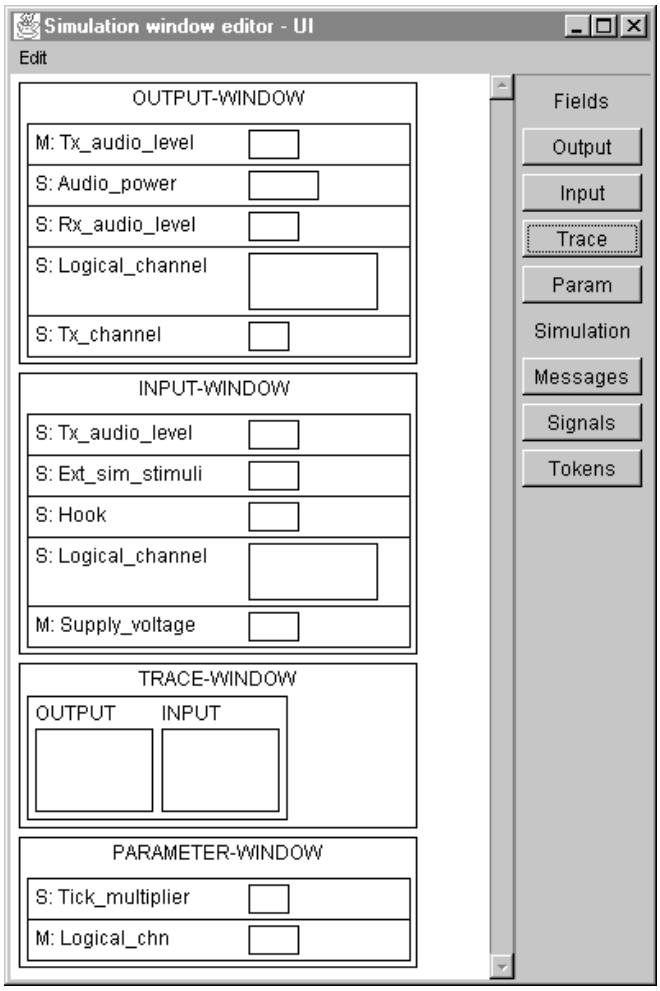
Taulukko 4. Ympäristöeditorin kuvakkeet.

	Simulointi-ikkuna. Ikkunan nimi näytetään tekstin ‘window’ edessä. Ikkunan tunnistenumero näytetään suluissa.
	Ulkoinen simulointiprosessi. Prosessin nimi näytetään vasemmassa yläkulmassa.
	Test controller unit.
	Testattava ohjelma.

4.5 Simulointi-ikkunan editointi

Kuvassa 12 on esitetty simulointi-ikkunan editointityökalu. Editorin piirtoalue on jaettu neljään ali-ikkunaan, nimeltään OUTPUT, INPUT, TRACE ja PARAMETER. Kun ikkunaan halutaan lisätä kenttä, ikkunaeditorin painikevalikosta painetaan sen ali-ikkunan painiketta, mihin kenttä halutaan lisätä. Kun painiketta on painettu, ilmestyy valintaikkuna, jossa määritellään uuden kentän ominaisuudet kuten nimi, pituus ja korkeus. Uusi kenttä lisätään aina ali-ikkunan pohjalle viimeiseksi kentäksi. Ali-ikkunan sisällä kenttien järjestystä voidaan vaihtaa valitsemalla kenttä osoittimella ja sitten raahaamalla kenttä haluttuun paikkaan.

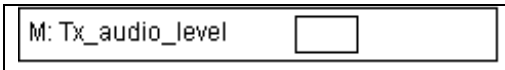
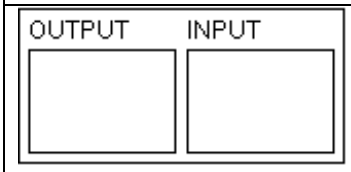
Trace-ali-ikkuna on erikoistapaus verrattuna muihin ali-ikkunoihin. Siinä voi olla vain output- ja input-kentät. Kuitenkin niin, että input-kenttä ei voi olla yksinään. Trace-ikkunan kenttien määrittelyyn tarkoitetussa dialogissa käyttäjä voi valita painonapeista millaisen yhdistelmän kentistä hän haluaa.



Kuva 12. Simulation Window Editor.

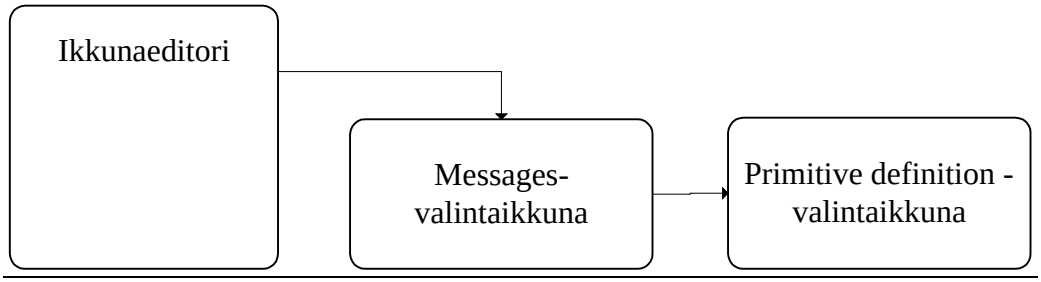
Taulukossa 5 on esitetty ikkunaeditorissa käytetyt kuvakkeet. Jokainen kenttä esitetään omana kuvakkeenaan, jossa esitetään kentän tyyppi, nimi ja koko. Valitun kentän tausta esitetään harmaana.

Taulukko 5. Ikkunaeditorin kuvakkeet.

	Viestikenttä. Vasemmalla sijaitseva iso M-kirjain ilmaisee, että kenttä on viestityyppiä. Mikäli kenttä on signaalityyppiä, se ilmaistaan kirjaimella S. Sisemmällä laatikolla ilmaistaan kentän pituus ja korkeus.
	Trace-kentät. Trace-kentät ovat erikoistapaus simulointi-ikkunan kentissä. Kenttiä voi olla vain kaksi, ouput ja input.

4.5.1 Viestien ja signaalien määrittely

Ikkunaeditorilla voidaan määrittää, kuinka kentistä lähtevät viestit ja signaalit välitetään simulointifunktioille. Määriteltäessä viestiä valitaan ensin kenttä, josta viesti lähtee simulointifunktiolle, ja painetaan *Messages*-painiketta, jolloin ilmestyy dialogi viestien määrittelyyn. Kuvassa 13 on esitetty, kuinka viestin määritysprosessi etenee. Koska samasta kentästä voi lähteä useita viestejä, viestien määrittely on tehty kaksivaiheiseksi. Ensimmäisessä vaiheessa eli *Messages*-dialogissa kontrolloidaan viestejä. Dialogissa voi valita tekeekö uuden viestin tai muuttaako tai tuhoaako vanhan viestimäärityksen. Varsinainen viestien määrittely tapahtuu dialogissa *Primitive definition*.



Kuva 13. Viestien määritysprosessi.

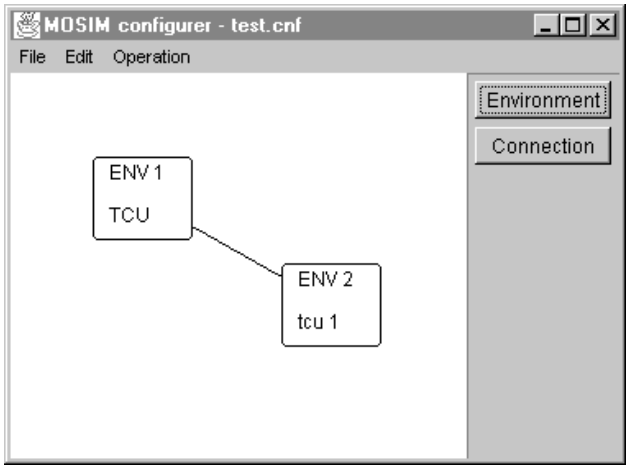
Signaalien määrityksessä täytyy myös valita ensin signaalikenttä, josta lähtevää signaalia määritellään. Kun kenttä on valittu ja painetaan *Signals*-painiketta, ilmestyy *Signal properties* -dialogi, jossa määritetään kuinka kentästä lähtevä signaali liitetään simulointifunktiolle. Koska signaalikentälle voidaan määrittää vain yksi signaalimääritys, ei tarvita erillistä välivaihetta kuten viestien määrittelyssä.

5. Käytettävyyden parantaminen ConfigToolissa

Tässä luvussa selvitetään, miten MOSIMin konfigurointia pyrittiin kehittämään Config-Toolissa. Päätarkoituksena on löytää tekijöitä käyttöliittymästä, jotka parantaisivat konfiguroinnin tekemistä. Lisäksi arvioidaan, kuinka visuaalisten kuvausten lisääminen on vaikuttanut käytettävyyteen. Käytettävyyttä parantavia tekijöitä etsitään käytännössä tutkimalla kolmea esimerkkiä konfiguroinnista. Ensiksi esitetään esimerkki verkon konfiguroinnista, toiseksi simulointi-ikkunan piirtäminen ja lopuksi tarkastellaan sanomien välittämistä keskeytyskäsittelijälle.

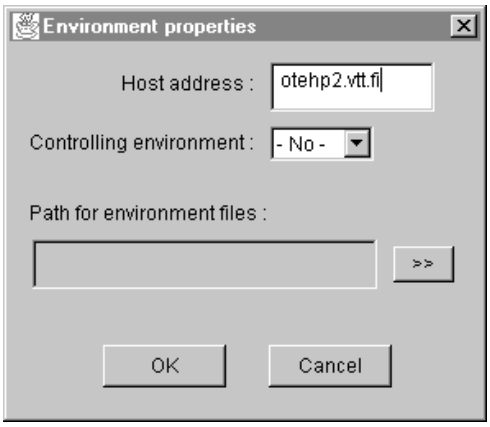
5.1 Verkon konfigurointi

Kuvassa 14 on esitetty esimerkki verkon konfiguraation tekemisestä. Esimerkissä on luotu kaksi ympäristöä, joiden tunnistenumerot ovat 1 ja 2. Ympäristö 1:ssä sijaitsee *Test Control Unit*, joka ohjaa myös ympäristöä 2. Ympäristöjen välillä on kommunikointikanava, jonka kautta ne voivat lähettää sanomia toisilleen.



Kuva 14. Esimerkki verkon konfiguroinnista.

Esimerkin konfiguraatio luodaan lisäämällä ensin kaksi ympäristöä. Ympäristön lisääminen tapahtuu painamalla *Environment*-painiketta, jolloin näytölle ilmestyy *Environment properties* -valintaikkuna (kuva 15). Valintaikkunaan täytetään tarvittavat parametrit, kuten osoite, missä ympäristö sijaitsee ja mahdollinen ohjaava ympäristö. Ohjaava ympäristö voidaan valita listasta, jossa on lueteltuna kaikki konfiguraatiossa olevat ympäristöt.



Kuva 15. Environment properties -valintaikkuna, jossa määritellään ympäristön parametrit.

Kun ympäristöt on lisätty, niiden välille voidaan luoda yhteys painamalla *Connection*-painiketta, jolloin ilmestyy *Connection*-valintaikkuna. Dialogissa valitaan yhteyden haltija ja kohdeympäristö. Määritys tapahtuu valitsemalla oikeat ympäristöt valintalistoista.

MOSIM Editor -tasolla käytettävät kuvakkeet aiheuttavat jonkin verran sekaannusta. Käyttäjät eivät heti huomaa kuvakkeissa olevien *tcu*-tekstien merkitystä. Usein ymmärrettiin, että kommunikointiväylää kuvaava viiva tarkoittaa ohjaavan ympäristön suhdetta ohjattavaan ympäristöön.

Jos kuvan 14 konfiguraatio tehtäisiin perinteisellä menetelmällä, käyttäjän pitäisi muodostaa kaksi tiedostoa, joiden sisältö on esitetty. Konfigurointi tapahtuisi kirjoittamalla komponentin nimi ja sen jälkeen tarvittavat parametrit komponentille. Esimerkiksi *mosim*-komponentille täytyy antaa parametriksi sen ympäristön tunnistenumero, jossa komponentti sijaitsee. Lisäksi voidaan antaa verkko-osoite, missä komponentti sijaitsee, kuten otehp2.vtt.fi. Jos ympäristöjä on useita ja jokin ympäristö ohjaa toista, ohjaavan ympäristön konfigurointitiedostoon täytyy sijoittaa maininta ohjattavien ympäristöjen *mosim*-komponenteista.

Kommunikointikanava lisätään sijoittamalla *link_mosim*-komponentti kumpaankin konfigurointitiedostoon. Komponentille annetaan parametriksi mm. yhteysnumero (connection id), joka täytyy olla kummassakin ympäristössä sama. Toinen *link_mosim*-komponenteista täytyy määrittää haltijaksi.

```
environment_id 1
mosim 1 otehp2.vtt.fi
link_mosim YES 1
2 elehp1.vtt.fi
environment 2.
mosim 2 elehp1.vtt.fi


environment_id 2
mosim 2 elehp1.vtt.fi
link_mosim NO 1 1
otehp2.vtt.fi
```

Kuva 16. Verkon konfigurointitiedostot ympäristöille 1 ja 2.

5.2 Käytettävyyden vertailu verkon konfiguroinnissa

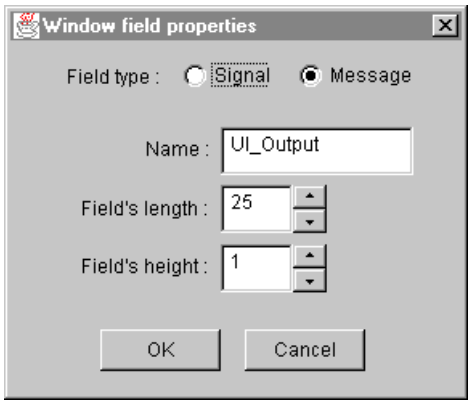
Esitetty esimerkki verkon konfiguroinnista osoittaa kuinka ConfigTool on helpottanut konfiguroinnin tekemistä verrattuna vanhaan manuaaliseen menetelmään. Vanhassa menetelmässä käyttäjä joutuu muistamaan komentoja, joilla selvitetään mitä komponentteja käytetään ja kuinka ne liitetään toisiinsa. Kuten kuvasta 16 näkee, on vanhan menetelmän tulkinta hankalaa.

Uudessa menetelmässä käyttäjän ei tarvitse muistaa konfigurointitiedostojen syntaksia. Lisäksi ConfigTool esittää tehdyt määritykset selkeästi ja ymmärrettävästi. Esimerkin konfiguroinnin teko on nopeata, koska suoritettavat toimenpiteet ovat lyhyitä.

5.3 Simulointi-ikkunan piirtäminen

Ikkunaeditorissa käyttäjä määrittelee simulointi-ikkunan ulkoasun. Ulkoasua varten käyttäjä määrittelee kentät, niiden leveyden, korkeuden, otsikon ja sijainnin.

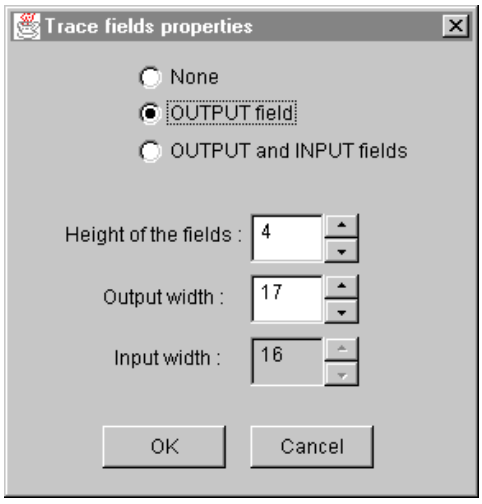
Uusi kenttä lisätään painamalla jotain neljästä Fields-painikkeesta, jolloin ilmestyy *Field properties* -dialogi. Siinä käyttäjä määrittelee kentän ominaisuudet. Kuvassa 17 on esitetty *Field properties* -dialogi, jota käytetään kenttien määrittelyyn output-, input- ja parameter-ali-ikkunoihin. Täytettäessä dialogia käyttäjän tarvitsee vain kirjoittaa kentän nimi, muut arvot hän voi syöttää käyttäen hiirtä.



Kuva 17. Dialogi kentän ominaisuuksien määrittelyyn.

Trace-ali-ikkunan kenttien määrittelyyn tarkoitettu dialogi on esitetty kuvassa 18. Trace-ali-ikkuna on erikoistapaus simulointi-ikkunan ali-ikkunoista. Ali-ikkunassa voi olla korkeintaan kaksi kenttää, output ja input, kolmella eri yhdistelmällä. Joko kumpaakaan kenttää ei ole, pelkästään output-kenttä tai sitten kumpikin. Koska trace kenttiä voi olla vain kolmella yhdistelmällä, voidaan trace-kentät määritellä radiopainikkeilla. Yhdistelmän valinnan lisäksi käyttäjän tarvitsee määritellä kenttien korkeus ja leveys. Kummallekin kentälle tulee sama korkeus, joten korkeuden määrittelyyn on varattu vain yksi syöttökenttä (*Height of the Fields*). Kummankin kentän leveys voidaan määritellä erikseen, joten kummallekin kentälle on varattu oma syöttökenttä leveyden määrittelyyn. Arvot voi syöttää joko näppäimistöllä tai kentän vieressä olevilla nuolipainikkeilla.

Koska output- ja input- kentistä voi olla vain tiettyjä yhdistelmiä, valintaikkunassa pyritään ohjaamaan käyttäjää niin, että valinnat tulevat oikein ja että käyttäjä ymmärtää mitä arvoja hänen tulee syöttää. Radiopainikkeista käyttäjä valitsee millaisen yhdistelmän hän haluaa output- ja input-kentistä ja valintaikkunan alaosassa olevia syöttökenttiä aktivoidaan käyttäjän valintojen mukaan.

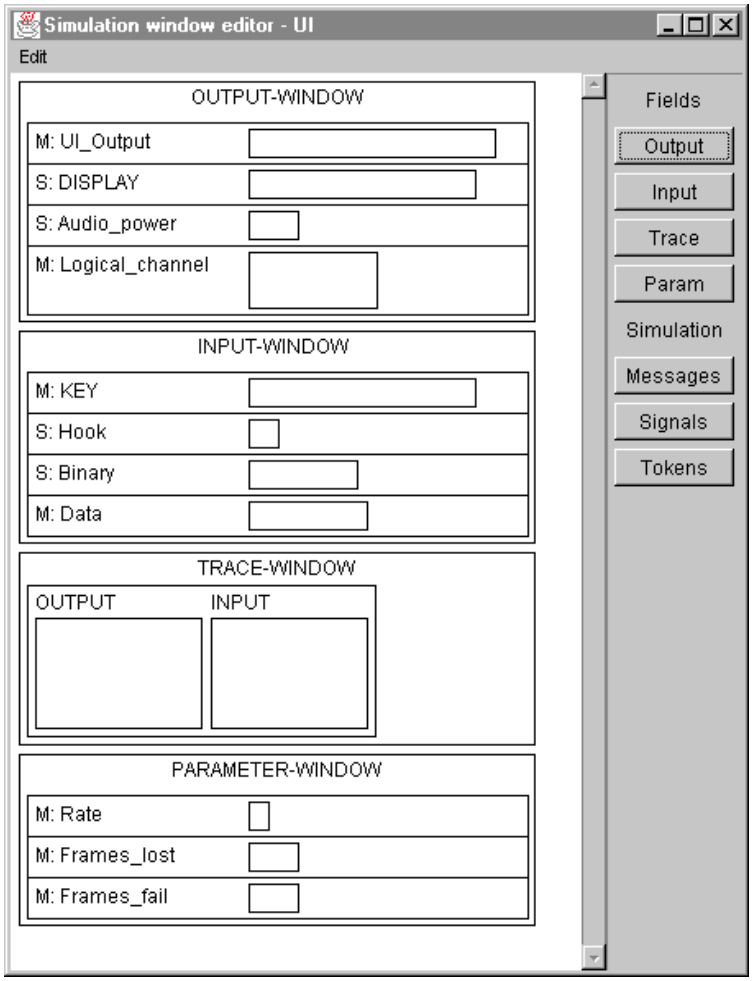


Kuva 18. Trace-kenttien määrittelyyn tarkoitettu valintaikkuna.

Kuvassa 19 on esitetty esimerkki eräästä simulointi-ikkunasta. Kentät ikkunaan on lisätty käyttäen kuvissa 17 ja 18 esitettyjä valintaikkunoita.

Aloitettaessa simulointi-ikkunan määrittely, ali-ikkunat ovat muutoin tyhjiä paitsi, parametri-ikkunassa valmiina oleva *Rate*-kenttä. Kun käyttäjä lisää kentän johonkin ali-ikkunaan, työkalu kasvattaa automaattisesti ikkunan raameja vastaavasti. Lisäksi kenttien kokoa kuvaavat laatikot sijoitetaan automaattisesti vasemman puoleisesta reunastaan linjaan.

Ikkunaeditori sijoittaa uudet kentät ali-ikkunan pohjimmaiseksi. Käyttäjä pystyy kuitenkin vaihtamaan kenttien paikkaa valitsemalla kentän ja siirtämällä sen oikeaan paikkaan. Kenttää voi liikuttaa vain pystysuorassa suunnassa ja ali-ikkunan sisällä. Kun kenttä on liukumassa jonkin toisen kentän päälle, alla oleva kenttä siirretään liikkuvan kentän alkuperäiselle paikalle ja liikkuva kenttä sijoitetaan vapautuneeseen paikkaan. Tällä tavoin saadaan estettyä, ettei käyttäjä siirrä kenttiä paikkoihin, joissa niiden ei ole mahdollista olla.



Kuva 19. Simulointi-ikkunan piirtäminen.

Kuvassa 20 on esitetty, kuinka kuvan 19 simulointi-ikkuna konfiguroitaisiin manuaalisesti vanhalla menetelmällä. Manuaalisella menetelmällä käyttäjän tulisi luoda tiedosto `<ikkunan_nimi>_mosim_window.h` johon hän määrittelisi ikkunan ulkoasun. Tiedostossa ikkuna määritellään käyttäen merkkejä %, !, & ja $. Jos kyseessä on signaali-ikkuna, käytetään huutomerkkiä kentän ilmaisemiseen ja vastaavasti viestikentälle $-merkkiä.

```
/*
%%%%%%%%%%%%%%%%%%%%%UI%%%%%%%%%%%%%%%%%%%%%
%                                              %
%&&&&&&&&&&&&&&&&OUTPUT-WINDOW&&&&&&&&&&&&&&&&%
% 1  UI_Output       $$$$$$$$$$$$$$$$$$$$$$$$ %
% 2  DISPLAY         !!!!!!!!!!!!!!!!!!!!!!   %
% 3  Audio_power     !!!!!                    %
% 4  Logical_channel $$$$$$$$$$$$             %
%                    $$$$$$$$$$$$             %
%&&&&&&&&&&&&&&&&&INPUT-WINDOW&&&&&&&&&&&&&&&&%
% 1  KEY             $$$$$$$$$$$$$$$$$$$$$$   %
% 2  Hook            !!!                      %
% 3  Binary          !!!!!!!!!!!              %
% 4  Data            $$$$$$$$$$$              %
%&&&&&&&&&&&&&&&&TRACE-WINDOW&&&&&&&&&&&&&&&&%
%    """""""OUTPUT"""""""   """""""INPUT""""""""   %
%    $$$$$$$$$$$$$$$$$$   $$$$$$$$$$$$$$$$$$   %
%    $$$$$$$$$$$$$$$$$$   $$$$$$$$$$$$$$$$$$   %
%    $$$$$$$$$$$$$$$$$$   $$$$$$$$$$$$$$$$$$   %
%&&&&&&&&&&&&&&&PARAMETER-WINDOW&&&&&&&&&&&&&&&%
% 1  Tick multiplier $$                         %
% 2  Rate            $$                         %
% 3  Frames_lost     $$$$$                      %
% 4  Frames_fail     $$$$$                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
*/
```

Kuva 20. Simulointi-ikkunan konfigurointi vanhalla menetelmällä.

Manuaalisessa menetelmässä käyttäjä joutuu itse huolehtimaan kenttien asemoimisesta ja linjaamisesta. Yleisiä ongelmia on ollut, että etukäteen on vaikea tietää mikä tulee olemaan pisin kenttä ja asemoida merkit sen mukaan. Esimerkin tapauksessa käyttäjä yleensä aloittaa UI_output -kentästä ja kun tullaan Logical_channel -kenttään, hän joutuu palaamaan alkuun ja siirtelemään kenttiä ja muita raameja vastaavasti oikeille paikoille.

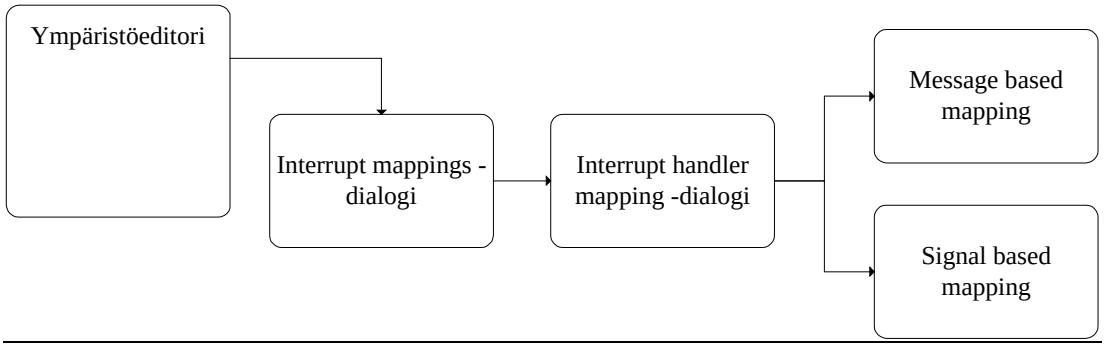
5.4 Käytettävyyden vertailu simulointi-ikkunan tekemisessä

Visuaaliseen ilmaisuun siitä, millaiselta konfiguroitava simulointi-ikkuna näyttää lopullisessa muodossaan, ConfigTool ei tuo paljon muutosta. Toisaalta työkalu automatisoi erittäin paljon, vapauttaen käyttäjän huomion tärkeämpiin asioihin. Myös muutosten teko on paljon helpompaa. Vanhalla menetelmällä käyttäjä joutuu itse naputtelemaan kaikki merkit. Esimerkiksi kenttien asettaminen samaan linjaan on paljon työläämpää.

5.5 Viestien ja signaalien liittäminen keskeytyskäsittelijöihin

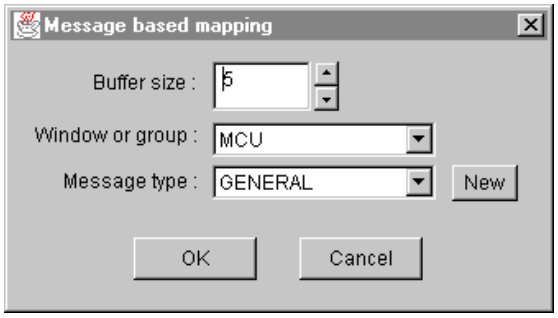
Sanomien liittäminen keskeytyskäsittelijöihin tapahtuu yleensä siinä vaiheessa, kun käyttäjä on määritellyt simulointi-ikkunat. Koska liitosten määritys tapahtuu valitsemalla valintalistoista sopivat ikkunat, ikkunaryhmät, ikkunakentät ja viestityypit, olisi suotavaa, että nämä komponentit olisivat olemassa.

Kuvassa 21 on esitetty, kuinka simulointi-ikkunoista tulevia viestejä ja signaaleja ohjataan keskeytyskäsittelijöille. Liittäminen alkaa ympäristöeditorista, josta aukeaa *Interrup mappings* -dialogi. Dialogista voidaan valita mille keskeytysfunktiolle sanomia halutaan ohjata. Kun funktio on valittu, dialogi *Interrupt handler mapping* aukeaa. Tästä dialogista käyttäjä voi määritellä kuinka sanomia välitetään valitulle funktiolle. Sanomien liittäminen voi tapahtua joko viestipohjaisesti tai signaalipohjaisesti. Kummallekin sanomatyypille on oma dialoginsa, jossa liitos määritellään.



Kuva 21. Viestien ja signaalien liittäminen keskeytyskäsittelijöille.

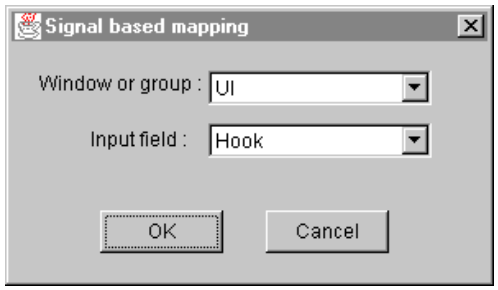
Kuvassa 22 on esitetty valintaikkuna, jolla määritellään viestipohjainen liittäminen keskeytyskäsittelijälle. Valintaikkunassa käyttäjä voi asettaa puskurin koon käyttäen nuolipainikkeita. Ikkuna tai ikkunaryhmä valitaan valintalistasta, samoin viestityyppi. Käyttämällä valintalistoja saadaan varmistettua, että valinnat ovat oikeita ja olemassa olevia.



Kuva 22. Viestipohjaisen liitoksen määritysikkuna.

Signaalipohjaisen liitoksen määrittelyikkunassa käyttäjää pyritään ohjaamaan myös valintalistoilla (kuva 23). Kun käyttäjä valitsee jonkin ikkunan, kentän valintalista

täytetään valitun ikkunan kentillä. Tällä tavoin saadaan ohjattua käyttäjä valitsemaan kenttä, joka todella on kyseisessä ikkunassa.



Kuva 23. Valintaikkuna signaalipohjaisen liitoksen määrittelyyn.

Kuvassa 24 on esimerkki kuinka manuaalisella menetelmällä on toteutettu muutama sanomien liittäminen keskeytyskäsittelijöihin. Esimerkissä on *app_ihandler*-funktiolle liitetty kaksi viestiä ja yksi signaali. Pollaavalle keskeytyskäsittelijälle on liitetty yksi viesti. ConfigToolilla tehtäessä vastaava konfiguraatio aloitettaisiin käynnistämällä uusi *Interrupt handler mapping* -valintaikkuna ja syöttämällä funktion nimeksi *app_ihandler*. Sen jälkeen käytettäisiin kuvien 22 ja 23 valintaikkunoita liitosten tekoon. Pollaava keskeytyskäsittelijä määriteltäisiin myös *Interrupt handler mapping* -valintaikkuna, mutta funktion nimen sijasta asetettaisiin *polling*-tarkistusruutu aktiiviseksi. Sen jälkeen käytettäisiin viestipohjaisen liitoksen määritysikkunaa liitoksen muodostamiseen.

ConfigToolilla käyttäjä ei pääse tekemään kirjoitusvirheitä. Käyttäjä pystyy valitsemaan vain ne komponentit, jotka todella ovat olemassa. Työkalun avulla käyttäjä huomaa paremmin, mitkä liitokset kohdistuvat millekin keskeytyskäsittelijälle. Valittavana olevien komponenttien esittäminen ei kuitenkaan ole kovin havainnollista. Komponentit esitetään käyttäjän antamien nimien perusteella ja annetut nimet eivät ole aina tarpeeksi kuvaavia. Käyttäjä ei esimerkiksi välttämättä muista mikä ikkuna oli nimeltään *MM*.

Manuaalisessa menetelmässä käyttäjillä on ollut tapana viitata komponentteihin suoraan niiden tunnistenumeroiden avulla. Tällöin konfigurointitiedostoista on muodostunut vaikealukuisia ja niiden ylläpidettävyys on kärsinyt. ConfigTool muodostaa generointivaiheessa standardimaiset vakiomäärittelyt tunnisteille ja näiden vakioiden käyttö takaa konfigurointien ylläpidettävyyden.

```
#define MOSIM_IH_1                          app_ihandler

#define MOSIM_IH_1_BUFFER_SIZE_MSG_1    5
#define MOSIM_IH_1_WINDOW_MSG_1         MCU_WINDOWS
#define MOSIM_IH_1_MSG_TYPE_1           GENERAL_MSG

#define MOSIM_IH_1_BUFFER_SIZE_MSG_2    2
#define MOSIM_IH_1_WINDOW_MSG_2         MM_MOSIM_WINDOW
#define MOSIM_IH_1_MSG_TYPE_2           GENERAL_MSG

#define MOSIM_IH_1_WINDOW_SIGNAL_1      UI_MOSIM_WINDOW
#define MOSIM_IH_1_ID_SIGNAL_1          UI_Hook_SLOT

#define MOSIM_IH_2                          0

#define MOSIM_IH_2_BUFFER_SIZE_MSG_1    1
#define MOSIM_IH_2_WINDOW_MSG_1         NETW_MOSIM_WINDOW
#define MOSIM_IH_2_MSG_TYPE_1           GENERAL_MSG
```

Kuva 24. Sanomien liittäminen keskeytyskäsittelijöihin vanhalla menetelmällä.

5.6 Aikavertailut

Kuten Booth ehdottaa, voidaan käytettävyyttä arvioida mittaamalla tehtävän suoritukseen kulutettua aikaa (Booth 1989, 120). ConfigToolin tapauksessa mittaus suoritettiin antamalla käyttäjälle konfigurointi-tehtäviä, jotka hän suoritti sekä manuaalisesti vanhalla menetelmällä ja sitten ConfigToolin avulla. Taulukossa 6 on esitetty mittauksen tulokset. Annetuissa tehtävissä käyttäjä teki samat konfiguraatiot kuin on aiemmin esitetty esimerkkeinä. Kummassakin menetelmässä käyttäjä tiesi ennalta mitä hänen tulee tehdä ja kuinka. Testitilanteessa mitattiin siis pelkästään konfiguroinnin suoraviivaiseen tekemiseen käytettyä aikaa.

Saatujen tulosten perusteella ConfigToolin avulla on nopeampi tehdä konfiguraatioita kuin manuaalisella menetelmällä. Manuaalisessa menetelmässä aika kuluu etupäässä turhauttavaan kirjoittamiseen ja käskyjen syntaksin tarkistukseen. ConfigToolissa toimintaa saadaan nopeutettua automatisoimalla turhauttavia toimintoja, kuten simulointi-ikkunan luomisessa.

Taulukko 6. Aikavertailut.

Tehtävä	ConfigTool	Manuaalinen
Verkon konfigurointi	1 min	3 min
Keskeystyskäsittelijöiden linkitys	1 min 17 sek	6 min 18 sek
Simulointi-ikkunan piirtäminen	3 min	10 min

Tehtävien suoritukseen käytetyn ajan mittaaminen ei kuitenkaan anna täysin selvää vastausta työkalun käytettävyyden paremmuudesta verrattuna vanhaan menetelmään. Käytännössä tehtävien suoritus erosi todellisesta tilanteesta siten, että käyttäjä tiesi etukäteen tarkasti millainen konfiguraatio hänen tulisi tehdä. Testitilanteessa käyttäjä eteni hyvin suoraviivaisesti pitämättä taukoja tai miettimättä tehtävää. Todellisessa tilanteessa käyttäjälle tulee taukoja, jolloin hän suunnittelee ja miettii tehtävää, sekä tekee korjauksia ja muutoksia. Muutosten tekeminen on nopeampaa ConfigToolin avulla, varsinkin simulointi-ikkunan ulkoasua tehdessä.

6. ConfigToolin käytettävyyden arviointi

ConfigToolille suoritettiin lyhyt käytettävyyden arviointi, jonka tarkoituksena oli selvittää, kuinka konfigurointityökalun käytettävyys on lisääntynyt. Arviointimenetelmänä käytettiin Nielsenin kehittämäm heuristista käytettävyysarviota ja Eisenstadin kehittämää arviointikehikkoa visuaalisille ohjelmointityökaluille.

ConfigToolin käytettävyyttä arvioitiin myös suorittamalla mielipidekysely. Kyselyyn valmistettiin lomake, jossa pyrittiin selvittämään Shackelin esittämiä asioita konfigurointityökalusta. Neljälle käyttäjälle annettiin konfigurointitehtäviä suoritettavaksi ConfigToolin avulla. Suoritettuaan tehtävän käyttäjät täyttivät kyselylomakkeen, joka on esitetty kuvassa 25.

1) Kuinka hyvin ConfigToolia oppi käyttämään? Mikä edisti oppimista ja mikä haittasi?

2) Oliko ConfigToolin käyttö turhauttavaa? Oliko jokin osio turhauttavampi kuin joku toinen? Mikä aiheutti turhautumista?

3) Tunsitko, että ConfigToolilla voisi työskennellä tehokkaasti? Kuinka arvioisit tehokkuutta verrattuna vanhaan tapaan konfiguroida MOSIMia?

4) Auttoiko ConfigTool hahmottamaan konfigurointia? Mitkä osiot auttoivat ja mitkä eivät? Mikä työkalussa auttoi hahmottamisessa?

5) Tunsitko onnistuvasi tehtävien suorittamisessa? Tunsitko epävarmuutta jonkin tehtävän onnistumisesta?

6) Luuletko, että ConfigToolilla pystyisi suorittamaan kaikki mahdolliset konfigurointitehtävät?

7) Oliko työkalua miellyttävä käyttää?

8) Millaisia virheitä käyttöliittymästä löytyi?

Kuva 25. Kyselylomake ConfigToolin arviointiin.

6.1 Nielsenin arviointimenetelmän tulokset

1) Järjestelmän tilan ilmeisyys.

Kun käyttäjä suorittaa jonkin toiminnon, toiminnon suoritus näkyy aina välittömästi näytöllä. Esimerkiksi kun lisätään jokin komponentti, näytölle lisätään kuvake esittämään uutta komponenttia.

Eräs kokenut manuaalisen konfigurointimenetelmän käyttäjä kommentoi, että tuntuu kuin käyttöliittymän takana tapahtuisi sellaista mitä ei näytetä. Tämä saattaa johtua siitä, että ConfigTool huolehtii automaattisesti monista tunnistenumeroista ja komponenttien välisten linkkien ylläpidosta. Manuaalisessa menetelmässä käyttäjä joutui itse huolehtimaan tunnistenumeroista ja linkeistä. ConfigTool ei ilmeisesti osoita kokeneille käyttäjälle, että asiat ovat hoidossa.

2) Järjestelmän ja todellisen maailman välinen yhteys.

ConfigToolin tapauksessa todellinen maailma tarkoittaa sitä käsitystä, minkä testausympäristön suunnittelijat luovat tilanteesta. Todellisia olemassa olevia komponentteja ovat ohjelmat, erilaiset näytöt ja portit. MOSIMin oppaissa ja koulutusmateriaalissa toimintaa ja komponentteja kuvataan erilaisilla kuvauksilla. ConfigToolin tulisi tukea kuvauksilla muodostettua käsitystä testausympäristön toiminnasta.

ConfigToolissa käytetyt termit ja nimikkeet on suurimmaksi osaksi otettu vanhasta manuaalisesta konfigurointimenetelmästä. Joissakin tapauksissa on kumminkin pyritty löytämään paremmin toimintoa kuvaavia nimikkeitä. Lisäksi on jouduttu kehittämään uusia toimintoja, jotta konfiguraatioon kuuluvien komponenttien väliset suhteet saataisiin kuvattua paremmin.

Käyttäjiä, jotka ovat kokeneita manuaalisen menetelmän käytössä, manuaalisen menetelmän termien käyttäminen auttaa ymmärtämään työkalun toimintaa. Uusille käyttäjille vanhat termit tuottavat ongelmia.

Uusi termi on luotu esimerkiksi kommunikointiväylän luomiseen kahden ympäristön välille. Vanhassa menetelmässä käyttäjä on luonut väylän *link_mosim*-käskyllä. ConfigToolissa väylä luodaan *Connection*-toiminnolla.

Myös sanomien linkittämisessä simulointifunktioille on pyritty käyttämään termejä, jotka kuvaavat paremmin määrityksen tarkoitusta. Aikaisemmin puhuttiin *primitiivien* määrittämisestä. ConfigToolissa määritys aloitetaan *messages*-toiminnolla.

3) Käyttäjän hallinta ja vapaus.

Useisiin valintaikkunoihin on toteutettu peruutus-toiminto (*Cancel*-painike), jolla käyttäjä voi perua ikkunaan tehdyt muutokset. Toimintoikkunoissa, joista käyttäjä voi kohdistaa ylläpitotoimintoja olemassa oleville määrityksille, ei ole peruutustoimintoja. Käyttäjä voi siis esimerkiksi tuhota jonkin määrityksen epähuomiossa eikä voi sitten palauttaa määritystä. Editoritasoilla ei ole toteutettu peruutus- eikä uudelleensuoritus toimintoja.

4) Yhdenmukaisuus ja standardimaisuus.

Valikkojen toiminta eri tasoilla on samankaltainen. Joka tasolla on *Edit*-valikko, josta löytyvät toiminnot komponentin poistamiselle ja muuttamiselle. *Edit*-valikossa on myös ikkunan sulkemistoiminto.

MOSIM- ja ympäristöeditoreista toisille tasoille siirtyminen tapahtuu kaksois-näpäyttämällä jotain kuvaketta. Tämä toimintoketju katkeaa ikkunaeditorissa, jossa kentän kuvakkeen näpäyttämisellä ei ole mitään toimintoa.

Muutamassa valintaikkunassa pyydetään määrittelemään komponentille *destination*, eli mihin mahdollinen toiminnan tulos kirjoitetaan. Kyse ei kumminkaan ole komponenteille yhteisestä määrityksestä, vaikka käytetäänkin samaa nimitystä. Tämä aiheutti jonkin verran hämmennystä käyttäjissä.

5) Virheiden ehkäisy.

Manuaalisessa konfigurointimenetelmässä virheistä suurin osa tuli kirjoitusvirheistä tai siitä, että käyttäjä unohti tehdä jonkun välttämättömän määrityksen. ConfigToolissa on pyritty ehkäisemään manuaalisessa menetelmässä esiintyvät virheet. Käyttäjän tekemät kirjoitusvirheet eivät pääse vaikuttamaan, koska komponenttien väliset liitokset toteutetaan työkalun ylläpitämillä sisäisillä viittauksilla.

6) Tunnistaminen mieluummin kuin muistaminen.

Tilanteissa, joissa komponentin määrittelyssä käytetään viittausta toiseen komponenttiin, on pyritty siihen ettei käyttäjän tarvitse muistaa komponentin nimeä tarkasti, vaan hän voi valita sen valintalistoista.

Työkalussa ei ole toteutettu minkäänlaista käytönaikaista opastusta.

7) Käytön joustavuus ja tehokkuus.

Työkalussa ei ole toteutettu minkäänlaisia oikopolkunäppäimiä esimerkiksi valikkotoiminnoille. Toimintoja ei voi oikoa, vaan kokeneidenkin käyttäjien on käytävä kaikki välitoiminnot lävitse. Käyttäjät eivät pysty räätälöimään usein käytettyjä toimintoja.

8) Esteettinen ja minimalistinen malli.

Jokainen valintaikkuna on suunniteltu niin, ettei niissä ole esillä kuin tarvittavat kentät kyseisen komponentin määrittelyyn. Useimpia kenttiä ei ole kuitenkaan välttämätöntä täyttää ja usein käyttäjät jättävätkin ne tyhjiksi. Joissakin valintaikkunoissa ei ole aina selvää mitkä kentät tulee täyttää ja mitä ei.

9) Käyttäjien avustaminen virheiden tunnistamisessa, määrittämisessä ja virheistä selviämisessä.

ConfigToolissa käytetään virheilmoituksia kahdessa eri tarkoituksessa. Konfigurointia muodostettaessa näytetään pienissä ilmoitusikkunoissa viestejä käyttäjälle. Viesteissä ilmoitetaan käyttäjälle mikäli hän yrittää käynnistää toiminnon, joka ei ole mahdollista, kuten tilanteessa, jossa käyttäjä yrittää ikkunaeditorissa käynnistää signaalin määrittelyn vaikka on valinnut viestikentän. Näytettävissä virheilmoituksissa kerrotaan mikä on vikana, mutta korjaustoimenpiteitä ei ehdoteta.

Kun tehtyä työkalulla tehtyä konfigurointia käännetään MOSIMin konfigurointi-tiedostoiksi, käyttäjälle näytetään käännösvaiheessa ilmenneet puutteet ja virheet. Virheilmoituksissa kerrotaan virheen sijainti aloittaen epätarkimmasta komponentista ja tarkentuen lopulta tarkimpaan komponenttiin, esimerkiksi ympäristö, ikkuna ja kenttä. Ilmoituksissa ei kerrota suoraan mitä tulee tehdä, mutta niissä kerrotaan puutteet ja virheet niin tarkasti, että käyttäjät huomaisivat nopeasti mitä tulee tehdä.

10) Opastus ja dokumentointi.

Opastusta ConfigToolissa ei ole toteutettu ollenkaan. Työkalulle on käyttöopas, jossa kerrotaan askel kerrallaan, kuinka kukin toiminto suoritetaan.

6.2 Yhteenveto Nielsenin arviointikriteerien toteutumisesta

Taulukossa 7 on esitetty yhteenveto Nielsenin esittämien kriteerien toteutumisesta ConfigToolissa. Taulukosta nähdään, että työkalussa on onnistuttu toteuttamaan hyvin virheiden ennaltaehkäisy. Käytettävyyttä haittaa jonkin verran peruutustoimintojen

puuttuminen. Erittäin heikkoa ConfigToolissa on oikopolkujen ja opastuksen puuttuminen.

Taulukko 7. Yhteenveto Nielsenin kriteerien toteutumisesta.

Kriteeri	Toteuttamistapa	Toteutumisaste
1)	Toimintojen välitön näyttäminen.	++
2)	Terminologia ja työkalun rakenne.	++
3)	Cancel-painikkeet valintaikkunoissa.	+
4)	Samanlaiset valikot ja siirtyminen editorien välillä.	++
5)	Kirjoitusvirheiden estäminen.	+++
6)	Valintalistat.	++
7)	Ei oikopolkuja.	--
8)	Vain tarvittavat kentät esillä.	++
9)	Virheilmoitukset ja ilmoitusikkunat.	++
10)	Käyttöopas. Ei opastusta ohjelmassa.	-

+++ = erittäin hyvin, ++ = hyvin, + = kohtalaisesti, - = puutteellinen, -- = hyvin puutteellinen, --- = ei ollenkaan

6.3 ConfigToolin onnistuneisuus visuaalisena työkaluna

Seuraavassa on arvioitu, kuinka ConfigTool täyttää Eisenstadin ja kollegoiden asettamat vaatimukset visuaaliselle työkalulle.

1) *Kuvaavuus*:

MOSIM Editor -tasolla ympäristöjen keskinäiset suhteet pystytään kuvaamaan melko hyvin. Ympäristöeditorissa esitetään kaikki ympäristöön kuuluvat komponentit havainnollisilla kuvakkeilla. Käyttäjä voi itse sijoittaa kuvakkeet niin, että ne esittävät

käyttäjän käsitystä ympäristön kokoonpanosta. Ikkunaeditorissa simulointi-ikkunan ulkoasu pystytään esittämään todellista vastaavassa muodossa.

Viestien ja signaalien linkittämiseen keskeytyskäsittelijöille ja simulointifunktioille työkalu ei tarjoa hyvää visuaalista esitystapaa. Määrittelyvaiheessa valintaikkunoissa on kyllä valintalistoja ja liukupalkkeja, joilla voidaan ohjata käyttäjä valitsemaan oikeat arvot. Arvot esitetään kuitenkin tekstuaalisessa muodossa, joiden kuvaavuus on kokonaan käyttäjän oman mielikuvituksen varassa.

2) *Käsiteltävyys*:

Jokaisella editoritasolla esitettävät komponentit ovat sijaintinsa suhteen käsiteltävissä. Käyttäjä voi valita komponentin ja liikuttaa sitä kuvaruudulla. Komponenttien muita ominaisuuksia joudutaan käsittelemään tarkoitukseen suunnitellun valintaikkunan kautta. Halutessaan muuttaa joitain ominaisuuksia käyttäjä joutuu valitsemaan komponentin ja sitten valitsemaan *Edit*-valikosta *Properties*-toiminnon.

Joitakin määrityksiä joudutaan muuttamaan pitkienkin dialogipolkujen takaa. Esimerkiksi viestipohjaista keskeytyslinkitystä pystyy muuttamaan vasta kolmannen dialogin takaa. Sanomien liittäminen keskeytyskäsittelijään tapahtuu ympäristöeditorissa, josta käynnistetään *Interrupt handler mappings* -dialogi. Dialogista voidaan sitten valita keskeytysfunktio, jonka sanomalinkityksiä käsitellään taas omassa dialogissaan. Tästä dialogista taas valitaan itse liitos, jota käsitellään *Message based mapping* -dialogissa.

Ympäristöeditorissa komponenttien liikuteltavuus piirtoalueella on oikeastaan turha ja raskaskäyttöinen. Työkalu voisi itse sijoittaa komponentit omille paikoilleen. Komponenttien sijainti toisiinsa nähden ei itse asiassa tarjoa mitään informaatiota. Käyttäjälle tulee vain turhaa työtä, kun hän siirtelee komponentteja syrjään muiden tieltä.

3) *Tulkittavuus*:

Painiketaulun painonappien nimikkeet eivät ole aina hyvin toimintaa kuvaavia. Toimintapainikkeissa on lyhyt tekstuaalinen nimi, josta ei aina käy ilmi mitä painike tekee. Esimerkiksi ikkunaeditorin *Output*-painikeen nimestä ei ilmene, että painike aloittaa prosessin, jolla lisätään kenttä Output-ikkunaan. Painikkeista tulisi ilmetä paremmin, että kyseessä on komponentin lisääminen.

4) *Kattavuus*:

Työkalulla pystytään tuottamaan laajojakin testausympäristöjä. Vaikka ympäristöjen määrä kasvaisi, ConfigToolin kuvausten monimutkaisuus ei kasva. Ainoastaan kuvakkeiden määrä näytöllä kasvaa. Kuitenkin suuri määrä alkioita näytöllä saattaa aiheuttaa sekaannusta.

Laajojen testausympäristöjen tuottaminen ConfigToolin avulla on helpompaa kuin vanhalla menetelmällä, koska työkalu huolehtii automaattisesti linkkien ja tunnistenumeroiden säilyttämisestä. Käyttäjä voi paremmin keskittyä itse ongelman ratkaisemiseen.

Käsiteltäessä laajoja tehtäviä voi ongelmia esiintyä lähinnä navigoitavuudessa.

5) *Näkyvyys*:

ConfigToolin periaatteena on, että kullakin editoritasolla näytetään vain sille tasolle liittyvät komponentit. Jos komponentteihin liittyen halutaan määritellä muita tekijöitä, käyttäjän täytyy avata dialogi jossa nämä asiat määritellään.

Käyttäjä ei pysty määrittelemään itse mitä komponentteja kussakin editorissa näytetään. Joissakin tilanteissa saattaisi olla hyödyllistä näyttää esimerkiksi, kuinka sanomat on liitetty keskeytyskäsittelijöihin ja simulointifunktioihin.

6) *Kytkentä*:

Joissakin dialogeissa komponenttien määritykset tehdään liittämällä komponentteihin muita komponentteja. Mikäli sopivaa liitettävää komponenttia ei löydy, käyttäjä voi käynnistää suoraan tämän komponentin määrittelyn, ilman että keskeyttää sillä hetkellä käynnissä olevaa toimenpidettä. Tällä tavalla määritellyt komponentit tulevat näkyviin myös työkalun muihin osioihin.

Esimerkiksi määriteltäessä viestejä on mahdollista liittää määritykseen simulointi-funktio. Itse simulointifunktioiden käsittely tapahtuu kuitenkin toisaalla työkalussa. Käyttäjän kannalta olisi kuitenkin ikävää keskeyttää viestin määritys sopivan simulointifunktion puuttumiseen ja palata toisaalle määrittelemään funktiota. Tämän vuoksi viestin määritykseen tarkoitetusta dialogista voi avata suoraan dialogin, jolla uuden simulointifunktion määritys onnistuu. Tämä funktio lisätään viestin määrittelyssä valittaviin simulointifunktioihin. Simulointifunktio on valittavissa määriteltäessä myöhemmin lisää viestejä.

Poistettaessa komponenttia, joka on liitetty toiseen komponenttiin, työkalu ei huomauta liitoksen olemassaolosta. Poistetun komponentin puutteen huomaa vasta, jos avaa ikkunan tai dialogin, jossa komponentti on liitetty.

Esimerkiksi jos poistaa jonkin simulointifunktion, ConfigTool ei huomauta, jos se on liitetty johonkin viesti- tai signaalimääritykseen. Avattaessa viestin määritysdialogi seuraavan kerran simulointifunktion kohdalla on teksti *no_token*.

7) *Navigoitavuus*:

Koska ConfigToolin kullekin eri tasolle käynnistetään oma ikkuna, näytölle tulee hetkessä useita ikkunoita. Tämän vuoksi navigoitavuuteen on jouduttu kiinnittämään erityistä huomiota. Jos esimerkiksi MOSIM Editor -tasolla kaksoisnäpäyttää jotain ympäristön kuvaketta ja tämän ympäristön editori on jo käynnistetty, kyseinen ympäristöeditori tuodaan päällimmäiseksi kuvaruudulla. Samanlainen toiminta on ympäristöeditorissa, kun halutaan käynnistää ikkunaeditori.

Käytännössä kuitenkin ikkunat peittävät toisia ikkunoita ja oikean ikkunan selaaminen esille on turhauttavaa. Haluamaansa ikkunaa on lisäksi vaikea tunnistaa muista saman tason ikkunoista. Eri tason editorit eroavat kuitenkin toisistaan niin paljon, että hierarkiatasot on helppo tunnistaa.

Päästäkseen määrittelemään joitakin komponentteja käyttäjä joutuu avaamaan useita dialogeja. Esimerkiksi linkitettäessä sanomia keskeytyskäsittelijöille, joudutaan avaamaan kolme dialogia. Kun jokin dialogi on auki, ei muita ikkunoita pysty käsittelemään.

8) *Täydellisyys*:

ConfigToolin tarkoituksena on ollut peittää MOSIMin konfigurointitiedostoissa tarvittavat syntaksit. Vanhassa menetelmässä on myöskin ollut eräiden tunnistenumeroiden laskemiselle mutkikkaita sääntöjä. Ei ole välttämätöntä, että konfigurointityökalu esittäisi tarkasti näiden tunnistenumeroiden luomisen. Esimerkiksi MOSIM Editor -tasolla työkalu huolehtii tunnistenumeron luonnista. Esittämällä vain tarvittavat asiat työkalun antaa käyttäjälle paremmat mahdollisuudet keskittyä itse suunnitteluun..

9) *Käännettävyys*:

Työkalulla on mahdollista kääntää kaikki tuotetut konfiguraatiot lopullisiksi MOSIMin konfigurointitiedostoiksi. Tuotettuja tiedostoja ei ole kuitenkaan mahdollista kääntää

takaisin työkalun ymmärtämään muotoon. Vanhojen konfiguraatioiden ylläpito ConfigToolin avulla ei siis onnistu.

6.4 Yhteenveto

ConfigTool pystyy toteuttamaan osan Eisenstadtin määrittelemistä ominaisuuksista kelvollisesti. MOSIMin konfiguroinnissa on kumminkin osioita joita ei pystytty toteuttamaan visuaalisesti, ja työkalun antama tuki testausympäristön suunnitteluun on kärsinyt tämän vuoksi. Esimerkiksi testattavalle sovellukselle tulevien sanomien ja signaalien yhdistäminen tapahtuu edelleenkin tekstuaalisia arvoja antamalla. Arvot kuitenkin syötetään dialogeissa ja useimmat arvot voidaan valita alasvetovalikoista, jolloin virheiden määrä voidaan minimoida (taul. 8).

Taulukko 8. Yhteenveto Eisenstadin vaatimusten toteutumisesta.

Kriteeri	Toteutustapa	Toteutumisaste
1) *Kuvaavuus*	Kuvakkeet, valintaikkunat, valintalistat ja liukupalkit.	++
2) *Käsiteltävyys*	Kuvakkeiden liikuteltavuus, muutokset *Edit*-toiminnon kautta.	+
3) *Tulkittavuus*	Tekstuaalinen tunnistaminen.	+
4) *Kattavuus*	Ei rajoituksia komponenttien määrässä.	+++
5) *Näkyvyys*	Näytetään vain tarvittavat komponentit.	++
6) *Kytkentä*	Tehdyt kytkennät näytetään tarvittaessa.	++
7) *Navigoitavuus*	Valittu editori tuodaan päällimmäiseksi näytölle.	+
8) *Täydellisyys*	Ei näytetä automaattisia toimintoja.	++
9) *Käännettävyys*	Piirroksista konfigurointitiedostoiksi.	-

+++ = erittäin hyvin, ++ = hyvin, + = kohtalaisesti, - = puutteellinen, -- = hyvin puutteellinen, --- = ei ollenkaan

7. Yhteenveto

Perinteisesti MOSIM-testausympäristö on konfiguroitu tekemällä manuaalisesti konfigurointitiedostot, joissa määritellään ympäristöön kuuluvat komponentit ja niiden väliset kytkennät ja sanomat. VTT Elektroniikassa toteutetussa MOTS2-projektissa tuotettiin visuaalinen konfigurointityökalu testausympäristön määrittelyyn. Tässä tutkielmassa tarkastellaan kuinka työkalun visuaalisuus on parantanut testausympäristön konfigurointia. Lisäksi selvitettiin, mitä visuaalisuus tarkoittaa ja miten sitä lisättiin konfigurointityökaluun. Tutkielman ongelmat voidaan esittää seuraavasti:

- Miten konfigurointityökalun käytettävyyttä voidaan parantaa?
- Miten konfigurointityökalun käytettävyys määritellään?
- Mitä visuaalisuus tarkoittaa konfigurointityökalun tapauksessa?
- Miten visuaalisuuden lisääminen konfigurointityökaluun tapahtuu?
- Paraniko MOSIM-konfigurointityökalun käytettävyys ConfigToolin avulla?

Konfigurointityökalun käytettävyyttä lisättiin kehittämällä siihen graafinen käyttöliittymä, jossa toteutetaan yleisiä suoran käsittelyn menetelmiä. ConfigToolissa käytetään ikkunoita jakamaan eri tason asiat erillisiksi kokonaisuuksiksi. Ikkunoissa taas on toteutettu valikkoja, joista käyttäjät voivat valita toimintoja. Valintaikkunoissa on käytetty ponnahdus- ja valintalistoja erilaisten valintamahdollisuuksien esittämiseen. Valikoilla ja valintalistoilla pystytään vähentämään käyttäjän tarvetta muistaa komentoja ja viitetunnuksia. Konfiguraation eri komponentteja esittämään kehitettiin erilaisia kuvakkeita, joita käyttäjä voi vapaasti liikuttaa näytöllä. Konfigurointityökalussa pyrittiin käyttämään visuaalisia kuvauksia, jotta käyttäjälle saataisiin paremmin kuvattua millainen konfiguraatiosta on tulossa ja hän pystyisi paremmin hahmottamaan tekemänsä konfiguraation.

Kyselytutkimukset osoittivat, että ConfigTool auttaa käyttäjiä hahmottamaan tekemänsä konfiguraation paremmin. Työkalun rakenteena toimiva malli konfiguroinnin etenemisestä helpotti hahmottamaan konfiguraation tekemistä. Erityisesti työkalun tasomaisuus ja asioiden ryhmittely oli käytettävyyttä edistävä seikka.

ConfigTool auttaa vähentämään käyttäjän kirjoitusvirheitä. Manuaalisessa konfigurointimenetelmässä käyttäjän täytyy osata syntaksi, jolla konfigurointi tapahtuu. ConfigTool vapauttaa käyttäjän syntaksin osaamisesta ja hän voi keskittyä konfiguraation tekemiseen.

Suoritettujen testien perusteella voidaan sanoa, paljonko ConfigTool nopeuttaa konfiguraatioiden tekoa. Erilaisten konfiguraatiotehtävien suorittamiseen käytetty aika pieneni vähintään kolmannekseen verrattuna perinteiseen manuaaliseen menetelmään. Lisäksi uudet käyttäjät kokivat, että ConfigToolin käyttö on nopeampi oppia.

Työkalun visuaalisuuden toteutuksessa on kuitenkin vielä puutteita. Kuvakkeiden esittämät asiat eivät ole itsestäänselviä ja aiheuttavat jonkin verran sekaannusta. Käytettävät nimikkeet ja termit eivät ole tarpeeksi kuvaavia. Varsinkin uusilla käyttäjillä on ongelmia termien kanssa.

Jotta ConfigToolin käytettävyyttä voidaan arvioida, täytyy määrittää mitä käytettävyydellä tarkoitetaan ConfigToolin tapauksessa. Mukaillen ISO:n määritelmää käytettävyydestä voidaan sanoa, että ConfigTool on hyvin käytettävä, jos käyttäjät on kykenevät tuottamaan konfiguraatioita tietyllä määrällä koulutusta, vaivaa ja ajan kulutusta. Konfiguraatioita on myös kyettävä tuottamaan tehokkaasti ja ylläpidettävästi.

Käyttöliittymän arviointiin voidaan käyttää samoja yleisiä arviointimenetelmiä kuin muihinkin ohjelmistoihin. Visuaalisten ohjelmointiympäristöjen arviointiin on Eisenstad kollegoineen on kehittänyt arviointikehikon, jonka tarkoituksena on selvittää, kattaako ohjelma kaikki tarpeelliset käytettävyyden osatekijät.

Jotta graafisesta käyttöliittymästä saataisiin mahdollisimman käytettävä, tulisi ohjelmiston kehitys tapahtua iteroivassa prototypointiprosessissa. Suunniteltavasta ohjelmistosta tuotetaan prototyyppi, jota arvioidaan sopivalla käytettävyyden arviointimenetelmällä. Saatuja tuloksia käytetään sitten ohjelmiston uudelleensuunnittelussa.

Lähteet

Booth, P. 1989. An introduction to human-computer interaction. Hove: Lawrence Erlbaum Associates Ltd.

Bragg, T. CASE and Visual Development Tools. American Programmer. November 1996,

Chang, S-K (ed). 1990. Principles of Visual Programming Systems. London: Prentice-Hall.

Eisenstad, M., Domingue, J, Rajan, T., Motta, E. 1990. Visual Knowledge Engineering. IEEE Transactions on Software Engineering. Vol. 16, no. 10, 1105-1197.

Honka, H. 1992. A simulation-based approach to testing embedded software. Espoo: Valtion teknillinen tutkimuskeskus. 118 s. (VTT Publications 124.) ISBN 951-38-4243-6

Huang, K-T. 1990. Visual Interface Design Systems. Teoksessa: Chang, S-K (ed). 1990. Principles of Visual Programming Systems. London: Prentice-Hall. S. 60-143.

Lund, A. M. & Tschirgi, J. E. 1991. Designing for People: Integrating Human Factors into the Product Realization Process. IEEE Journal on Selected Areas in Communications. Vol. 9, no. 4., 496-500.

Nielsen, J. 1994. Usability Inspection Methods. New York: John Wiley & Sons.

Preece, J. 1993. A Guide to Usability, Human Factors in Computing. Wokingham: Addison-Wesley Publishing Company.

Redmond-Pyle, D. & Moore, A. 1995. Graphical User Interface Design and Evaluation. Hertfordshire: Prentice Hall International.

Shu, N. C. 1998. Visual Programming. New York. Van Nostrand Reinhold Company.

Wozniewich, A. The Future of Visual Development. American Programmer. November 1996.