VTT TIEDOTTEITA – MEDDELANDEN – RESEARCH NOTES 1759

CAD-suunnittelutiedon hyödyntäminen mekatronisen laitteen kunnonvalvonnassa

Panu Korpipää
VTT Elektroniikka

VALTION TEKNILLINEN TUTKIMUSKESKUS
ESPOO 1996

ISBN 951-38-4914-7
ISSN 1235-0605

JULKAISIJA – UTGIVARE – PUBLISHER

Valtion teknillinen tutkimuskeskus (VTT), Vuorimiehentie 5, PL 2000, 02044 VTT
puh. vaihde (90) 4561, telekopio 456 4374

Statens tekniska forskningscentral (VTT), Bergsmansvägen 5, PB 2000, 02044 VTT
tel. växel (90) 4561, telefax 456 4374

Technical Research Centre of Finland (VTT), Vuorimiehentie 5, P.O.Box 2000, FIN–02044 VTT, Finland
phone internat. + 358 0 4561, telefax + 358 0 456 4374

Tekninen toimitus Leena Ukskoski

VTT OFFSETPAINO, ESPOO 1996

Korpipää, Panu. CAD-suunnittelutiedon hyödyntäminen mekatronisen laitteen kunnonvalvonnassa [Utilisation of CAD engineering information for condition monitoring of mechatronic systems]. Espoo 1996, Valtion teknillinen tutkimuskeskus, VTT Tiedotteita – Meddelanden – Research Notes 1759. 65 s. + liitt. 3 s.
UDK 681.3:621.3:681.518.5
Avainsanat design automation, design knowledge, fault diagnosis, machine automation, object-oriented product modeling

TIIVISTELMÄ

CAD-suunnittelutiedon ja -tietämyksen hyödyntäminen tarjoaa lupaavia mahdollisuuksia mekatronisen laitteen kunnonvalvonnan ja vikadiagnostiikan näkökulmasta. CAD-suunnittelutiedon käyttö laitteen ohjausjärjestelmässä vaatii järjestelmällisiä tiedon esitystapoja sekä siirto- ja muokkausmenetelmien kehittämistä.

Tavoitteena oli CAD-sähkösuunnittelun ja ohjelmistosuunnitteluprosessin välisen rajapinnan tiedonsiirron tehostaminen automatisoimalla suunnittelutiedon siirto ja muokkaus CAD-sähkökuvista C-kielisiksi tietorakenteiksi. Tietorakenteita hyödynnettiin maanalaiseen kaivaukseen tarkoitetun kallioporauslaitteen reaaliaikaisessa kunnonvalvontaohjelmassa, joka piti suunnitella ja toteuttaa. Kunnonvalvontaohjelmalta vaadittiin selkeä ja yksinkertainen tiedon välittäminen hajautettujen liityntämoduulien ja toimilaitteiden tiloista, antureiden arvoista ja sähköjärjestelmän rakenteesta.

Työvaiheiden automatisoimiseksi suunnitteluprosessien välillä otettiin käyttöön suunnittelutiedolle yhteinen tietokanta. CAD-mallien ja tietokannan välinen tiedonsiirto voidaan toteuttaa CAD-järjestelmien tarjoamien sovelluskehitys-ympäristöjen avulla. Tietokannan ja ohjelmistosovelluksen välistä tiedonsiirtoa varten kehitettiin ohjelma, joka tuottaa tietokannan tietojen perusteella C-kieliset tietorakenteet sovellusta varten. Tietorakenteita hyödyntävä reaaliaikainen kunnonvalvontasovellus suunniteltiin simulaatiomallien avulla. Sulautettava kunnonvalvontaohjelma testattiin kallioporauslaitteen ohjausjärjestelmää vastaavassa testilaitteessa.

Tuloksena demonstroitiin menetelmä CAD-sähkösuunnittelutiedon siirtämiseksi automaattisesti sulautettavien ohjelmistosovellusten käyttöön. Kallioporaus-laitteen kunnonvalvontaohjelmaa voidaan käyttää vikahavainnoinnissa. Työ pohjautui perinteisiin CAD-suunnittelutiedon mallinnusmenetelmiin. Laajamittai-semmassa suunnittelutiedon ja -tietämyksen kuvaamisessa suositellaan käytettäväksi kehittyneitä oliopohjaisia tuotemallinnusmenetelmiä.

Korpipää, Panu. CAD-suunnittelutiedon hyödyntäminen mekatronisen laitteen kunnonvalvonnassa [Utilisation of CAD engineering information for condition monitoring of mechatronic systems]. Espoo 1996, Valtion teknillinen tutkimuskeskus, VTT Tiedotteita – Meddelanden – Research Notes 1759. 65 p. + app. 3 p.
UDK 681.3:621.3:681.518.5
Avainsanat design automation, design knowledge, fault diagnosis, machine automation, object-oriented product modeling

ABSTRACT

Utilisation of CAD engineering information for a mechatronic system provides promising possibilities for condition monitoring and fault diagnosis. Use of CAD engineering information in the control system requires systematic ways of presenting information and developing transforming and transfering methods.

The aim was to make transfer of information between the CAD electrical engineering and software engineering processes more efficient by automating information transfer from CAD models into C language data structures. Data structures were used by a real-time condition monitoring program for underground mining drilling rigs. The requirements for the program to be designed and implemented were to clearly and simply supply information about states of decentralized interface modules and actuators, values of sensors, and the structure of the electric system.

For automating the activities between these engineering processes, a common database was used for the design information. Application development environments of CAD systems can be used to transfer the information between CAD models and the database. For transfering information between the database and the application program, a Windows application was developed. This application generates the C language data structures based on information from the database. The real-time condition monitoring program which utilises these data structures, was designed using simulation models. The monitoring program to be embedded was tested in a simulator for the control system of the drilling rig.

A method for automatic transfering of CAD engineering information into application programs was demonstrated as the result. The condition monitoring program for the drilling rig can be used in fault detection. This work was based on traditional CAD engineering information modeling methods. It is recommended to apply advanced object-oriented product modeling methods in larger scale utilisation of design information and knowledge.

ALKULAUSE

Tämä diplomityö on tehty VTT Elektroniikassa Tamrock Oy:n toimeksiannosta.

Esitän parhaat kiitokseni työn valvojalle professori Petri Pullille ja toiselle tarkastajalle tutkimusprofessori Veikko Seppäselle.

Kiitän työni ohjaajina toimineita fil. tri. Matti Kurkea ja fil. maist. Eila Niemelää asiantuntevasta opastuksesta. Työn ohjelmistosuunnitteluun osallistui dipl. ins. Jarmo Mäkelä Tamrock Oy:stä. Hänelle esitän kiitokseni hyvästä yhteistyöstä ja käytännöllisistä neuvoista.

Lopuksi vielä kiitokset työtovereilleni ja Piritalle.

Oulussa 30.1.1996

Panu Korpipää

SISÄLLYSLUETTELO

LYHENTEIDEN JA MERKKIEN SELITYKSET

680x0	Motorolan prosessoriperhe
A/D	Analog to Digital, analogisesta digitaaliseen
ANSI	American National Standards Institute, standardointiorganisaatio
ASCII	American Standard Code for Information Interchange, aakkosnumeeristen merkkien koodausjärjestelmä
ASIC	Application Specific Integrated Circuit, sovelluskohtainen integroitu piiri
C	Ohjelmointikieli
C++	Oliopohjainen C
CAD	Computer Aided Design, tietokoneavusteinen suunnittelu
CAE	Computer Aided Engineering, tietokoneavusteinen insinöörityö
CAN	Controller Area Network, hajautetun järjestelmän tiedonsiirtoverkko
CE	Concurrent Engineering, rinnakkainen insinöörityö
CPU	Central Processing Unit, keskusyksikkö
D/A	Digital to Analog, digitaalisesta analogiseen
DDE	Dynamic Data Exchange, dynaaminen tiedonsiirto
DOS	Käyttöjärjestelmä
ER	Entity-Relationship, relaatiomalli
GUI	Graphical User Interface, graafinen käyttöpaneeli
I/O	Input/Output, tulo/lähtö
LISP	List Processing, ohjelmointikieli
AutoLISP	AutoCAD:n LISP-pohjainen ohjelmointikieli
OOCAD	Object Oriented Computer Aided Design -projekti
OS9	Reaaliaikakäyttöjärjestelmä
OXF	Object Exchange File, tiedonsiirtoformaatti
PC	Personal Computer
PROM	Programmable Read Only Memory, ohjelmoitava lukumuisti
RAM	Random Access Memory, luku- ja kirjoitusmuisti
RS-232	Sarjaliikennestandardi, fyysinen liitäntä
RS-422	Sarjaliikennestandardi, fyysinen liitäntä
SA/SD	Structured Analysis/ Structured Design, rakenteinen suunnittelu
STEP	Standard for the Exchange of Product Model Data, standardi tuotetietojen mallintamiseen ja tiedonsiirtoon
TAS	Tamrock Automation System, yleiskäyttöinen ohjausjärjestelmä porauslaitteiden automatisointiin
TIESU	Tietämyksen integrointi ja sulauttaminen prosessihallinnassa -projekti
UNIX	Käyttöjärjestelmä
VHDL	Very High Speed Integrated Circuit Hardware Description Language, digitaalisten piirien kuvauskieli
VME	VERSAmodule Eurocard, Motorolan kehittämä väyläratkaisu

1 JOHDANTO

Moniteknologisten mekatronisten laitteiden kehittyessä ja monimutkaistuessa niiden suunnittelutiedon ja -tietämyksen määrä on lisääntynyt. Järjestelmän kompleksisuus aiheuttaa suuria ongelmia vikatilanteissa, mikäli suunnittelutietämystä ei ole kunnonvalvonnan saatavilla. Ongelmat ovat synnyttäneet tarpeen suunnittelutiedon ja -tietämyksen järjestelmälliseen esittämiseen ja sulauttamiseen ottaen huomioon tiedon käyttömahdollisuudet laitteen koko elinkaaren aikana, varsinkin kun laitteiden sulautettujen tietokonejärjestelmien kapasiteetit ovat kasvaneet ja hinnat laskeneet.

Pyrkimyksenä on korkean abstraktiotason rakennekuvausten, kuten CAD-mallien, sisältämän tiedon automaattinen muuntaminen sellaiseen muotoon, että sitä voidaan hyödyntää mekatronisen laitteen sulautetussa ohjausjärjestelmässä. Suunnitteluprosessin tehostumisen lisäksi tällä tavoin vähennetään ihmisen tekemien virheiden määrää. Ylläpidettävyys paranee huomattavasti, koska CAD-malleihin tehtyjä muutoksia ei enää tarvitse erikseen ohjelmoida sulautettavaan ohjelmistoon. Laitteiden toimitusaikojen lyhentäminen ja asiakaskohtaistaminen ovat myös paremmin mahdollisia suunnittelun osa-alueiden välisen tiedonsiirron kehittyessä.

Tässä työssä tarkastellaan mekatronisen laitteen CAD-suunnittelutiedon ja -tietämyksen esittämistapoja kunnonvalvonnan ja vikadiagnostiikan näkökulmasta. Lisäksi tutkitaan suunnittelutiedon siirtämis- ja muokkaamismahdollisuuksia laitteeseen sulautettavien sovellusohjelmien osaksi.

Työn tavoitteena oli CAD-sähkösuunnittelun ja ohjelmistosuunnittelun välisen rajapinnan tiedonsiirron tehostaminen automatisoimalla aikaisemmin käsin tehtyjä työvaiheita. Niitä olivat CAD-sähkösuunnittelussa liityntätiedon muokkaus kytkentälistoiksi ja ohjelmistosuunnittelussa sulautettavan sovellusohjelman C-kielisten tietorakenteiden ohjelmointi kytkentälistojen perusteella.

CAD-malleista C-kielisiksi tietorakenteiksi muokattuja tietoja tuli hyödyntää maanalaiseen kaivaukseen tarkoitetun kallioporauslaitteen reaaliaikaisessa kunnonvalvontaohjelmassa, joka piti suunnitella ja toteuttaa. Kunnonvalvontaohjelmalta vaadittiin selkeä ja yksinkertainen tiedon välittäminen käyttäjälle hajautettujen liityntämoduulien ja toimilaitteiden tiloista, antureiden arvoista ja sähköjärjestelmän rakenteesta. Tässä oli tarkoituksena käyttää hyväksi CAD-sähkömalleista saatua tietoa hajautettujen liityntämoduulien sekä toimilaitteiden ja osajärjestelmien tulojen ja lähtöjen erityyppisistä ominaisuuksista. Tiedon esittämistavan tuli olla hierarkkinen ja tiedot piti sovittaa kohdejärjestelmän graafisen käyttöpaneelin näytölle. Lisäksi oli suunniteltava kunnonvalvontaohjelmalle helposti ymmärrettävä ja looginen käyttöliittymä ohjeineen.

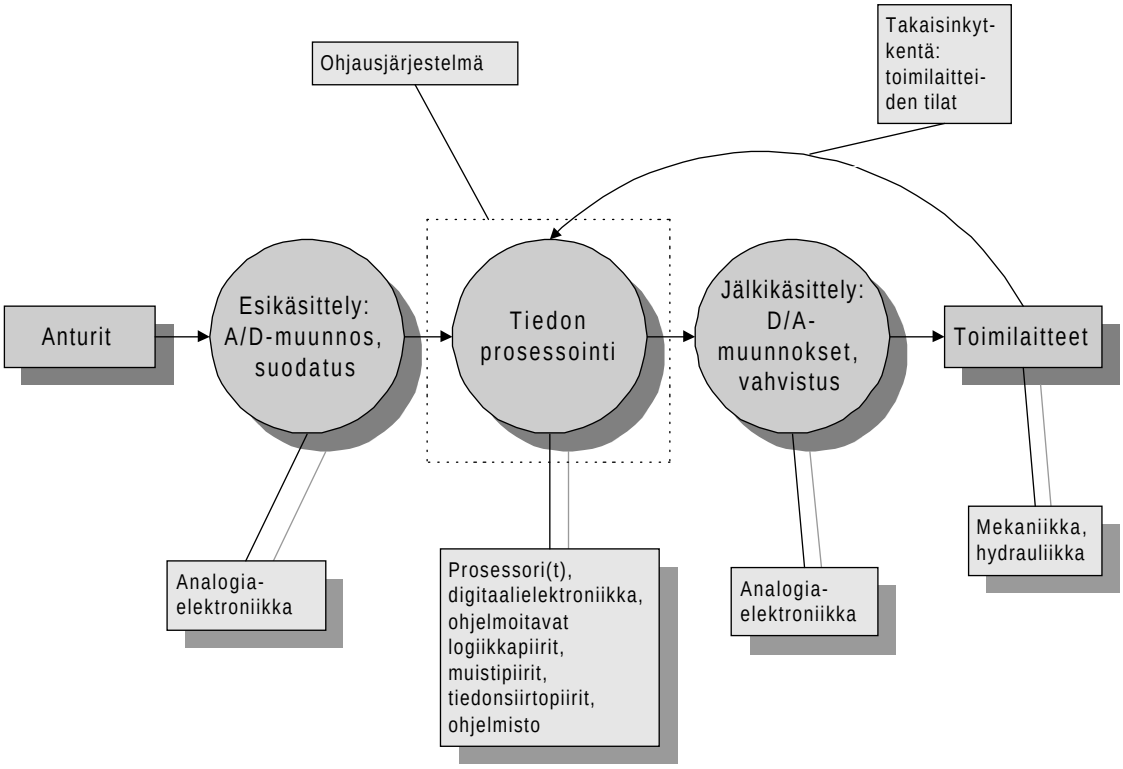
2 MONITEKNOLOGINEN MEKATRONINEN LAITE

Nimeä “mekatroninen laite” voidaan käyttää kuvaamaan laajaa erilaisten automaattisten koneiden joukkoa. Robotit, kaivinkoneet, nosturit, hissit, kallioporauslaitteet, lentokoneet ja sukelluskellot kuuluvat kaikki tähän sovellusalueeseen. Jopa nykyaikaista henkilöautoa aktiivijousituksineen ja lukkiutumattomine jarruineen voidaan tietyssä mielessä pitää mekatronisena laitteena.

2.1 MEKATRONISEN LAITTEEN OMINAISUUDET

Mekatroninen laite vastaanottaa signaaleja ympäristöstään, prosessoi ne ohjausjärjestelmässään ja reagoi niihin esimerkiksi liikkein. Reagoinnin ympäristön tapahtumiin ja ärsykkeisiin tulee lähes aina tapahtua reaaliajassa. Esimerkiksi kallioporauslaitteen vikadiagnostiikkajäjestelmän reagointiajan kriittiseen vikaan pitäisi olla korkeintaan kymmeniä millisekunteja vian vaikutusten rajoittamiseksi. Joissakin tapauksissa vaaditaan vielä paljon nopeampaa vastetta, kuten esimerkiksi lentokoneen “fly-by-wire”-ohjauksessa [1]. Tarpeet luonnollisesti vaihtelevat sovelluksen mukaan, mutta keskimäärin vasteajat asettavat erityisiä vaatimuksia järjestelmän ohjauselektroniikalle ja -ohjelmistolle. Useat mekatroniset laitteet ovat lisäksi jatkuvakäyttöisiä eli edellyttävät suurta toimintavarmuutta. Myös toimintaympäristö riippuu sovelluksesta, mutta useimmissa tapauksissa mekatroninen laite joutuu toimimaan ankarissa ja vaikeissa olosuhteissa. [1]

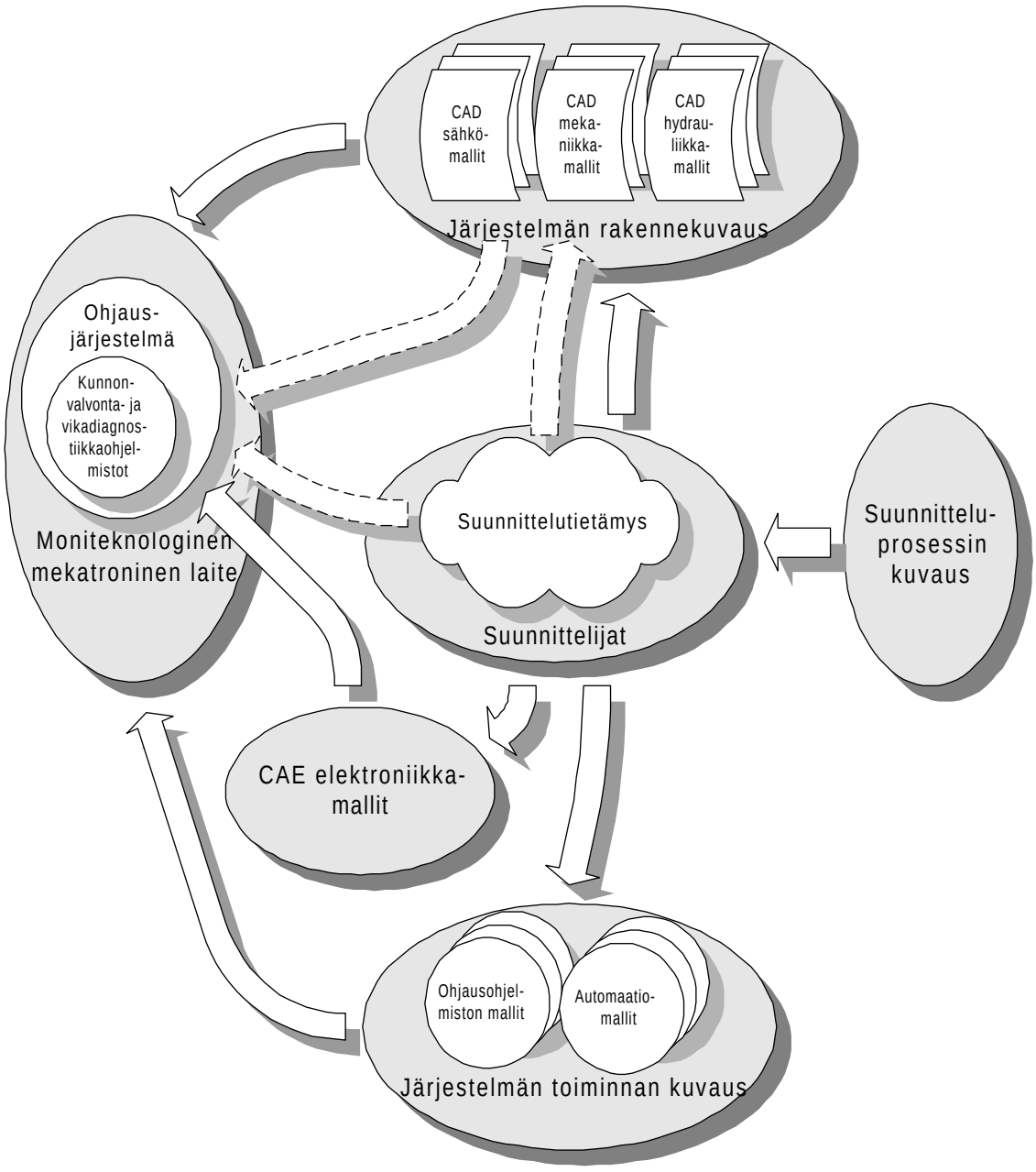
Mekatroniset järjestelmät ovat rakenteeltaan monimutkaisia. Ne koostuvat moniteknologisista komponenteista, jotka eroavat toisistaan merkittävästi sekä toiminnaltaan että rakenteeltaan. Ne ovat yhdistelmä ohjelmistoa, prosessoreja ja digitaalielektroniikkaa, analogiaelektroniikkaa, mekaniikkaa ja hydrauliikkaa. Ohjausjärjestelmän muodostavat prosessointiyksikkö apupiireineen, joihin kuuluvat muistipiirit ja tiedonsiirtoyksiköt, sekä ohjausohjelmisto, systeemiohjelmisto ja toimilaitteiden ajurit. Ohjelmisto-osat on sulautettu osaksi ohjausjärjestelmää. Sulautetun ohjelmiston toiminnot toteutetaan mm. ASIC-piireillä tai erilaisilla ohjelmoitavilla logiikkapiireillä tai sijoittamalla ohjelmisto esimerkiksi PROM-muistiin. Analogiaelektroniikkaa käytetään pääosin mekaniikan ja hydrauliikan liittämiseksi ohjausjärjestelmään. Ympäristön herätteiden ja toimilaitteiden tilojen pohjalta ohjausjärjestelmässä tehdään päätökset toiminnoista, jotka suoritetaan mekaniikan ja hydrauliikan avulla (kuva 1). [2, 3, 4]



Kuva 1. Mekatronisen laitteen rakenteen ja toiminnan periaate.

2.2 MEKATRONISEN LAITTEEN SUUNNITTELU

Moniteknologisen laitteen suunnittelu koostuu useista vaiheista ja osa-alueista, jotka eroavat toisistaan esitystavoiltaan ja suunnitteluperiaatteiltaan. Järjestelmän rakennetta kuvataan ensisijaisesti CAD-mallien avulla. Esimerkiksi mekaniikka-, hydrauliikka- ja sähköjärjestelmien rakenne esitetään CAD-mallien avulla. Toimintojen kuvaamiseen käytetään monenlaisia kuvaustapoja: automaatiomallit [1], ohjelmiston vuokaaviot [5] ja ohjauselektroniikan VHDL-kuvaukset [6] ovat mahdollisia malleja järjestelmän eri osien toiminnalle. Osa-alueiden liityntöjen esittämistä varten käytetään rajapintakuvauksia. Tässä työssä käsitellään tarkemmin CAD-sähkösuunnittelun ja ohjelmistosuunnittelun rajapintakuvausta. Kaikkien mallien, dokumenttien ja muiden kuvausten lisäksi suunnittelijoilla on laitteesta ja sen osista tietämystä, jota ei välttämättä tallenneta ollenkaan. Mekatronisen laitteen tehokkaan kunnonvalvonnan ja huollon kannalta olisi välttämätöntä saada yhdistettyä suurin osa tästä tiedosta ja tietämyksestä käyttökelpoiseen muotoon laitteen ohjausjärjestelmään, kuva 2. [7, 8]



Kuva 2. Moniteknologisen mekatronisen laitteen suunnittelussa käytettävät tärkeimmät kuvaustavat.

Suunnitteluprosessia hidastavat yleensä eri osa-alueiden välinen kommunikointi sekä tiedon yhdistäminen ja siirto lopputuotteeseen. Kommunikointiongelmat vaativat ylimääräisiä palavereja ja tuottavat lisää dokumentteja, joilla tietoa pyritään siirtämään alueelta toiselle. Osittaisena ratkaisuna tähän voisi ainakin CAD-suunnittelun osalta olla eri osa-alueiden suunnittelumenetelmien ja esitystapojen standardointi. Ongelmia aiheuttavat lisäksi kuvassa 2 katkoviivanuolilla esitetyt tietämyksen taltiointi ja siirto sekä rakennekuvausten siirto laitteen ohjausjärjestelmään.

2.2.1 Suunnitteluprosessin kehittäminen

Mekatronisia laitteita valmistavalle vientiyritykselle kova kansainvälinen kilpailu aiheuttaa paineita suunnitteluprosessin kehittämiseksi ja tehostamiseksi. Suunnittelu-, testaus- ja valmistusaikoja on pakko saada lyhennettyä. Rinnakkaisen insinöörityön (CE, Concurrent Engineering) soveltamisen [9] ja standardoinnin lisäksi pyritään karsimaan "turhia" välivaiheita suunnitteluprosessin eri alueilla automatisoimalla työvaiheita. Tavoitteena on yleisesti abstraktiotason nostaminen suunnittelussa mahdollisimman havainnolliselle asteelle. Tässä työssä on suunnitteluprosessin osalta tutkittu CAD-sähkösuunnittelun ja ohjelmisto-suunnittelun rajapinnassa tapahtuvien työvaiheiden tehostamista tiettyjen toimintojen automatisoinnilla.

2.3 VAATIMUKSET KUNNONVALVONNALLE

Mekatronisten laitteiden yhä monimutkaistuessa vikadiagnostiikkaan ja kunnonvalvontamahdollisuuksiin on kiinnitettävä yhä enemmän huomiota. Tärkeimpiä tavoitteita kunnonvalvonnalle ovat laitteen käytettävyyden lisääminen ja huoltokustannusten minimoiminen. Mekatronisen laitteen korkea käytettävyys sekä huollon ja korjauksen ennakoitavuus ja helppous antavat kilpailuedun laitteen valmistajalle. Järjestelmien kohonnut automaatioaste ja käytön turvallisuus asettavat omat lisävaatimuksensa kunnonvalvonnalle. [2]

Kehittyneen kunnonvalvonta- ja vikadiagnostiikkajärjestelmän toteuttaminen edellyttää suunnittelutiedon hyödyntämistä. Laitteen monimutkaistuessa CAD-mallien, huolto-ohjeiden, käsikirjojen ja muiden dokumenttien määrä kasvaa niin suureksi, että tiedon rakenteinen esittäminen ja siirtäminen laitteen ohjausjärjestelmän osaksi tulee lähes välttämättömäksi.

2.3.1 Laitteistovaatimukset

Mekatronisen laitteen ohjausjärjestelmältä vaaditaan kunnonvalvonnan ja vikadiagnostiikan tiedonkäsittelyprosessien takia entistä enemmän tehoa. Tämä ei kuitenkaan ole ongelma, koska prosessoritekniikan kehitys on nopeaa. Suunnittelu- ja muiden dokumenttien integrointi ohjausjärjestelmään vaatii muistikapasiteettia, mikä lisää jonkin verran laitteen komponenttikustannuksia. Suurimmaksi käytännön ongelmaksi muodostunee kuitenkin yhteys käyttäjään. Kunnonvalvonta ja vikadiagnostiikka ovat parhaimmillaan, kun tiedot voidaan esittää graafisesti käyttäjälle. Graafiset näyttölaitteet ovat kuitenkin vielä toistaiseksi kookkaita, suhteellisen kalliita ja herkkiä vikaantumaan vaikeissa olosuhteissa.

3 MEKATRONISEN LAITTEEN KUNNONVALVONTA

3.1 KÄYTTÄJÄN TOIMINTA VIKATILANTEESSA

Oletetaan tilanne, jossa monimutkaiseen mekatroniseen laitteeseen tulee vika eikä kunnonvalvonta- tai vikadiagnostiikkaominaisuuksia ole sisällytetty sen ohjausjärjestelmään. Käyttäjä havaitsee vian oireet lampuista tai mittareista, jotka näyttävät kriittisiä arvoja ilman mitään aputietoja. Oireiden tulkinta riippuu tällöin kokonaan käyttäjän kyvyistä ja kokemuksesta. Tulkinta voi johtaa esimerkiksi seuraavanlaisiin toimintoihin:

- Varmistutaan, ettei kyseessä ole väärä hälytys.
- Kysytään apua kokeneemmalta käyttäjältä, jos mahdollista.
- Lähdetään etsimään lisätietoja CAD-malleista, muista suunnitteludokumenteista tai huolto-ohjeista, mikäli dokumentit löytyvät.
- Jos ratkaisua ei löydy, pyydetään paikalle huoltomies, joka joutuu suorittamaan todennäköisesti edellä mainitun tehtävän.

Tavoitteena on huomattava ajansäästö ja turvallisempi toiminta edellä mainittuun tilanteeseen verrattuna. Tällöin vaatimuksena on suunnittelutiedon ja -tietämyksen, käsikirjojen ja muiden dokumenttien löytyminen online-tietona laitteen ohjausjärjestelmästä. Tämä yhdistettynä valvonta- ja vikadiagnostiikkaominaisuuksiin antaa mahdollisuuden laitteen tehokkaaseen ja turvalliseen käyttöön.

3.2 KUNNONVALVONNAN TEHTÄVÄT

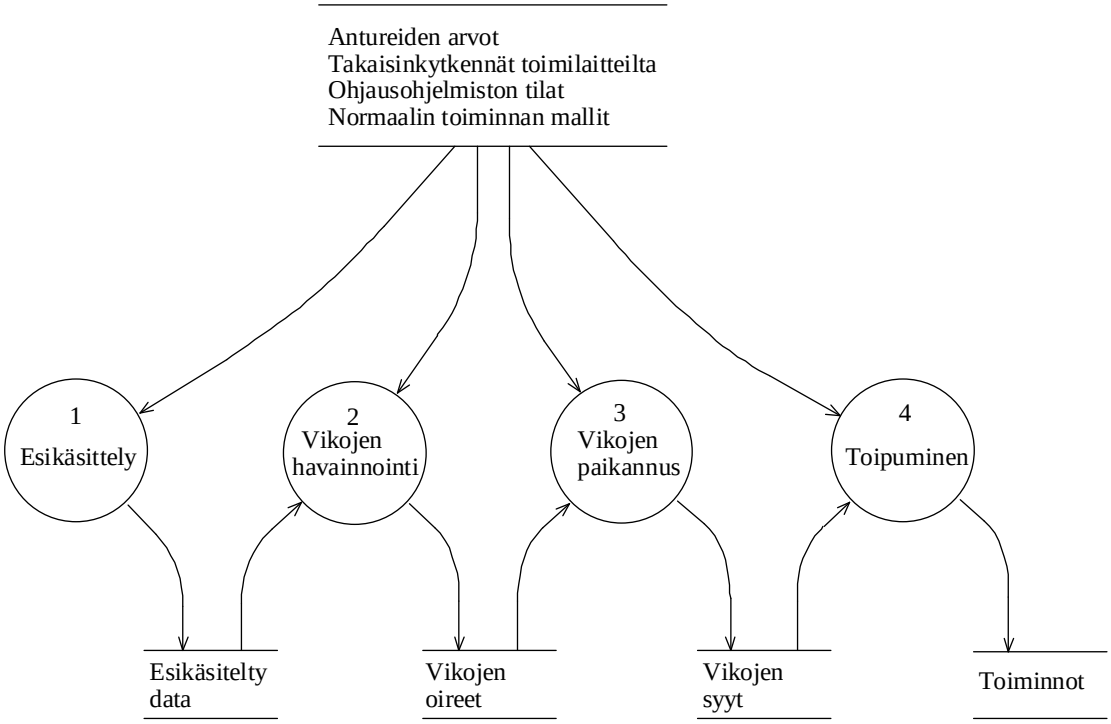
Mekatronisen laitteen käytettävyys on tärkeimpiä asioita käyttäjälle laitteen ollessa toiminnassa. Kunnonvalvonta- ja vikadiagnostiikkajärjestelmien tulee tarjota tietoa vikaoireista ja niiden mahdollisista syistä joko automaattisesti tai esittämällä vian paikannusta helpottavaa tietoa. Tämän lisäksi käyttäjää kiinnostavat vikojen mahdolliset vaikutukset laitteen toimintaan ja tarvittavat korjaustoimenpiteet. Laitteen toimiessa käyttäjä voi tarvita vastauksen seuraaviin kysymyksiin:

- Toimiiko laite normaalisti?
- Onko kyseessä vika tai vikaan myöhemmin johtava oire?
- Kuinka vakava vika on?
- Voiko vian korjata heti ja, jos voi, miten?
- Tarvitaanko huoltoa ja, jos tarvitaan, millaista?
- Kuinka kauan laitetta voidaan käyttää ennen huoltoa?
- Kuinka kauan mahdollinen huolto kestäisi?

Lisäksi järjestelmän tulisi tarjota käyttäjää päätöksenteossa avustavaa tietoa. Esimerkiksi graafisesti esitettynä aputieto tukee tehokkaasti vikadiagnostiikka-prosessia eri vaiheissa. [2]

3.3 VIKADIAGNOSTIIKKA

Mekatronisen laitteen kunnonvalvonta ja vikadiagnostiikka ovat erillisiä käsitteitä, joskin edellisen voidaan ajatella olevan osa jälkimmäistä. Vikadiagnostiikalla tarkoitetaan yleisesti laajempaa prosessia, johon kuuluu anturitiedon esikäsittely, vikojen havainnointi ja paikantaminen sekä kriittisistä vioista toipuminen. Kuva 3 esittää vikadiagnostiikkaprosessin tehtäväjakoa [2].



Kuva 3. Vikadiagnostiikkaprosessin jako osatehtäviin.

3.3.1 Esikäsittely

Anturitiedon esikäsittelyllä tarkoitetaan laitteen ympäristöstä ja ohjausjärjestelmästä kerätyn suuren tietomäärän suodattamista ja muokkaamista jatkokäsittelyn mahdollistamiseksi. Esikäsittelyllä on kaksi tehtävää, tarkistaa tulojen järkevyys ja suorittaa tarvittavat laskutoimitukset vikojen havainnointia varten. [2]

3.3.2 Vikojen havainnointi

Vikojen havainnointi voi olla joko automatisoitua tai käyttäjän suorittamaa. Vikahavainnoinnin ollessa automaattista diagnostiikkajärjestelmä kykenee päättelemään, milloin vikatilanne on kyseessä. Toisin sanoen järjestelmä kykenee selvittämään vian oireet käyttäjän asemesta. Voidaan esimerkiksi mallintaa laitteen normaalia toimintaa ja käyttäytymistä, jolloin vikatilanteet nähdään poikkeuksina normaalitoiminnasta. Tämä voidaan toteuttaa vertailemalla antureiden tiloja niille määrättyihin laillisiin arvoihin. Joka kerta ohjausjärjestelmän tilan vaihtuessa kaikkien antureiden arvot tarkastetaan. Mikäli jokin anturi on tietyllä hetkellä väärässä tilassa, kyseessä on vikatilanne. Järjestelmä tarkkailee myös analogisten antureiden arvoja sekä niiden muutoksia, muutosnopeuksia ja -suuntia mahdollisten vikojen etsinnässä. Nämä arvot sisältävät tyypillisesti myös tärkeää aputietoa käyttäjälle. Vikahavainnoinnin tuloksena voisi olla esimerkiksi “paine on liian korkea” tai “kytkinanturin arvo on väärä”. [2]

3.3.3 Vikojen paikantaminen

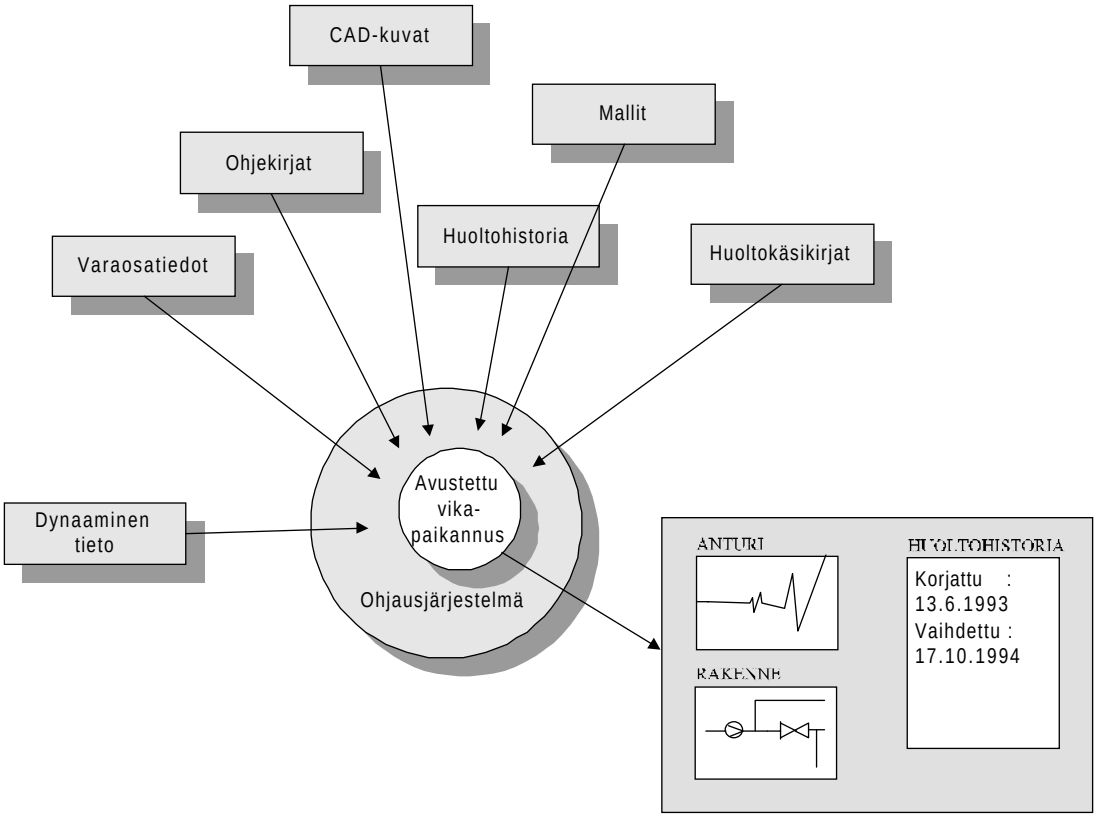
Vikapaikannus tarjoaa tietoa vikojen mekanismeista, aiheuttajista ja syistä. Vian paikantamisen voi käynnistää käyttäjä tai järjestelmä itsenäisesti havaittuaan vian oireen. Tietty vika voi aiheuttaa monenlaisia eri oireita. Monissa tapauksissa vian suora paikantaminen on mahdotonta puuttuvan tiedon vuoksi. Lisäksi samaan vikatilanteeseen saattaa olla syynä monta erilaista aiheuttajaa ja vikojen leviäminen voi vaikeuttaa alkuperäisen vikalähteen etsintää. Näistä syistä täysin automatisoitu paikannus ei välttämättä ole paras vaihtoehto. Paremman tuloksen antaa yhdistelmä automaattista ja avustettua vikapaikannusta. Automaattinen paikannus on tarpeen, jos vika on kriittinen ja vaatii välitöntä huomiota. Tällöin vikaan voidaan reagoida heti muuttamalla järjestelmän ohjausta tai ilmoittamalla siitä välittömästi käyttäjälle. Muussa tapauksessa käyttäjä voi toimia vian paikantajana järjestelmän antaessa ohjeita ja aputietoa. Vikapaikannuksen tuloksena voisi olla esimerkiksi “anturi on rikki”, “hydrauliikkaletku vuotaa” tai “suodatin on tukossa”. [2]

3.3.4 Toipuminen

Diagnostiikkaprosessin päätarkoituksena on perinteisesti ollut vikojen aiheuttajien löytäminen. Kaikki käyttäjät eivät kuitenkaan osaa tulkita oikein vikapaikannuksen tuloksia tietääkseen, miten täytyy toimia. Toipumisella on kolme perustehtävää: pakotettujen toimintojen suorittaminen, toimintojen ehdottaminen käyttäjälle ja huoltomiehen tai käyttäjän avustaminen vikatilanteen selvittämiseksi. Pakotettuja toimintoja tarvitaan, kun kyseessä on kriittinen vika, josta voisi aiheutua ihmisille tai laitteistolle vahinkoa. Muulloin voidaan käyttää tilanteeseen liittyvän aputiedon ja ohjeiden esittämistä käyttäjälle tai huoltomiehelle. Toipuminen voidaan toteuttaa yhdistämällä vikojen syyt oireisiin ja tarvittaviin korjaaviin toimenpiteisiin. [2]

3.4 AVUSTETTU VIKAPAIKANNUS

Aputiedon tarkoituksena on tarjota käyttäjälle kaikki tarpeellinen tieto vikapaikannusprosessin helpottamiseksi. Järjestelmän mallit ja niihin liittyvä tietämys, käyttöohjeet, huolto-ohjeet, vikahavainnoinnin tieto ja rakennekuvaukset ovat esimerkkejä vikapaikannuksessa tarvittavasta aputiedosta (kuva 4).



Kuva 4. Avustettu vikapaikannus.

Käyttäjien ja huoltomiesten tiedot laitteen rakenteesta, toiminnasta ja käyttäytymisestä sekä vikatilanteista vaihtelevat huomattavasti. Järjestelmän tarjoama aputieto on keino tämän tiedonpuutteen tasoittamiseksi. Pelkkä tiedon näyttäminen käyttäjälle ei tosin riitä, vaan esitetyn tiedon täytyy olla havaittuihin vikaoireisiin liittyvää. [2]

3.5 APUTIEDON YHDISTELY

Nykyisissä järjestelmissä suurin osa vikapaikannuksen aputiedosta on huoltokäsikirjoissa, käyttöohjekirjoissa ja CAD-malleissa. Käyttäjän täytyy etsiä monista eri lähteistä toimintaohjeita, varaosatietoa, toiminnan kuvauksia, rakennetietoa ja tietoja vaihdetuista osista. Siirtämällä aputieto osaksi laitteen ohjausjärjestelmää (kuva 4) saavutetaan monia huomattavia etuja:

- Kaikki tieto on helposti saatavilla yhdessä paikassa.
- Staattinen ohjekirjatieto voidaan kytkeä dynaamiseen käyttöhetken tietoon.
- Näytettävä tieto voidaan valita järjestelmän tilan perusteella.

Jotta aputietoa voitaisiin täysin hyödyntää vikapaikannuksessa, täytyy olla käytettävissä dynaamista järjestelmätietoa ja rakennetietoja. Nämä tulisi liittää huoltotietoihin yhtenäiseksi kokonaisuudeksi [2]. Rakennetietojen hyödyntäminen on mahdollista toteuttaa esimerkiksi CAD-tuotemallinnuksen avulla. Tuotemallinnusta kevyempi vaihtoehto joidenkin rakennetietojen siirtoon on käyttää tässä työssä toteutettua tietorakenteiden tuottamismenetelmää. Menetelmässä siirretään ja muokataan CAD-tietoa automaattisesti kunnonvalvontaohjelman käyttöön ja samalla käyttäjälle aputiedoksi.

3.6 KUNNONVALVONTA OSANA VIKADIAGNOSTIIKKAA

Kunnonvalvonnan voidaan ajatella olevan osa vikadiagnostiikkaprosessia. Tarkemmin sanottuna kunnonvalvontaa voidaan pitää kyseisen prosessin toisena tehtävänä eli vikahavainnointina. Tämän työn toteutusosassa tehdyllä kunnonvalvontaohjelmalla on toteutettu käyttäjän suorittama vikahavainnointi. Automatisoituun havainnointiin verrattuna erona on se, että järjestelmä ei itse tunnista vikaoireita, vaan käyttäjän täytyy ne päätellä. Tämä asettaa tietysti vaatimuksia käyttäjän pätevyydelle. Seuraamalla kytkinantureiden tiloja käyttäjän pitäisi pystyä päättelemään, toimiiko järjestelmä tietyssä tilanteessa oikein. Analogia-antureiden arvoja ja niiden muuttumista seuraamalla käyttäjän olisi kyettävä erottamaan mahdolliset vikaoireet.

Tällaisen menettelyn ongelmana on nimenomaan se, että kaikki käyttäjät eivät ole yhtä asiantuntevia laitteen toiminnan suhteen. Epäkohta voidaan ratkaista käyttämällä käyttäjän suorittamaa kunnonvalvontaa automatisoidun vikahavainnoinnin rinnalla. Näin on tehty esimerkiksi tämän työn kohdejärjestelmässä. Automatisoitu vikahavainnointi toimii jatkuvasti kunnonvalvontaohjelman ollessa valinnainen rinnakkainen prosessi. Tällä tavoin voidaan yhdistää molempien menettelyjen hyvät puolet. Automatisoitu vianhaku on varmatoiminen laitteen normaalista poikkeavan toiminnan havaitsemisessa. Käyttäjä taas saattaa huomata anturien arvoissa ja laitteen toiminnassa muutoksia,

jotka diagnostiikkajärjestelmältä voivat mahdollisesti jäädä havaitsematta, ja tällöin puuttua asiaan.

4 TIEDON JA TIETÄMYKSEN ESITTÄMINEN

4.1 SUUNNITTELUTIETÄMYS

Suunnittelutietämyksellä tarkoitetaan kaikkea sitä tietämystä [10, 11], joka johtaa määrättyyn suunnittelutulokseen. Tulos voi olla esimerkiksi mekatroninen laite. Asiakkaan tarpeet, tuotemäärittelyt, tuotteen toteutuksen ratkaisuvaihtoehdot perusteluineen, suunnittelijoiden päätökset ja niiden perustelut, hylätyt ratkaisut, osien ominaisuudet, osavalintojen perusteet, laitteen ominaisuudet ja toiminta ovat kaikki osa suunnittelutietämystä.

Suunnittelutietämys voidaan jakaa kahteen pääosaan, tuoteläheiseen ja suunnitteluläheiseen [12]. Tuoteläheinen suunnittelutietämys pyrkii selittämään suunniteltavaa laitetta kuvaamalla suunnittelijoiden käsitystä laitteen toiminnasta. Laitteen rakenteen esittäminen ja havainnollistaminen sekä osakokonaisuuksien ja komponenttien ominaisuuksien, toiminnan ja rakenteen selittäminen kuuluvat myös tähän kategoriaan. Suunnitteluläheinen tietämys puolestaan käsittelee suunnitteluprosessia, päätöksiä, niiden perusteita ja suunnitteluhistoriaa [12]. Kunnonvalvonnan kannalta merkitystä on kuitenkin lähinnä tuoteläheisellä suunnittelutietämyksellä, johon tässä työssä rajoitutaan.

4.2 SUUNNITTELUTIETÄMYKSEN KERUU

Mekatronisten laitteiden tuoteläheisen suunnittelutietämyksen kuvaamisessa ja esittämisessä CAD-mallit ovat pääosassa. Läheskään kaikkea kunnonvalvonnassa tarvittavaa tietämystä ei kuitenkaan nykyään tallenneta eikä kuvata. Esimerkiksi seuraavanlaiselle komponentteja koskevalle tietämykselle olisi käyttöä kunnonvalvonnassa ja vikadiagnostiikassa:

- osan rakenteelliset ja toiminnalliset liitynnät
- osakohtaiset mahdolliset viat ja oireet
- vikaantumisherkkyys ja vikaantumistodennäköisyys
- osan fyysisen sijainnin kuvaaminen muuhun laitteeseen nähden
- vian sattuessa tarvittavat korjaustoimenpiteet tai tarkistukset
- huoltotoimenpiteet.

Suunnitteluprosessi kuvaa tällä hetkellä osan edellä mainitusta tietämyksestä, mutta tieto on yleensä hajallaan ja vaikeasti hyödynnettävissä kunnonvalvontaa ajatellen. Luonnollisin ja loogisin tapa kuvata, järjestää ja tallentaa puuttuva suunnittelutietämys olisi tietysti juuri CAD-järjestelmien avulla.

Tilanne ei kuitenkaan ole näin yksiselitteinen. Kaikkea tuotekohtaistakaan suunnittelutietämystä ei välttämättä voida kuvata CAD-malleilla. Lisäksi

tietämyksen tallentaminen kasvattaa suunnittelijoiden työmäärää. Tämä on ollut myös erilaisten tietämysjärjestelmien kehittämistä vaikeuttava tekijä: Tietämyksen keruu erillisenä hankkeena yksittäistä sovellusta varten on yksinkertaisesti liian kallista ja työlästä. [12]

VTT:n TIESU-projektissa (Tietämyksen integrointi ja sulauttaminen prosessihallinnassa) on pohdittu edellä mainittua ongelmaa. Projektissa esitettiin seuraava kannanotto: “Ainoa realistinen vaihtoehto näyttää olevan se, että tietämys kirjataan siellä, missä se syntyy, ja että se voidaan siirtää helposti (automaattisesti?) muihin vaiheisiin.” Lause tarkoittaa käytännössä sitä, että suunnittelutietämyksen keruun tulisi tapahtua luonnollisena osana varsinaista suunnittelutyötä [12]. Tämä edellyttää tietysti suuriakin muutoksia nykyisiin suunnittelukäytäntöihin ja -kulttuuriin sekä suunnittelusääntöjen tarkentamista ja yhdenmukaistamista. Työkaluilta vaaditaan monipuolisia kuvausmahdollisuuksia sekä automatisoitua tiedonsiirtoa eri osa-alueiden ja suunnittelun työvaiheiden välillä.

4.3 CAD-SUUNNITTELUN KEHITYKSESTÄ

Tietokoneavusteisen suunnittelun työvälineet ja teknologiat ovat viimeisen vuosikymmenen aikana kehittyneet huomattavasti. Niiden ominaisuuksia on pyritty kehittämään suunnittelun eri osa-alueiden työvaiheiden tehostamiseksi ja automatisoimiseksi. Alkujaan CAD-järjestelmien kehitys on lähtenyt liikkeelle piirtämisestä. Alkuvaiheessa tekniikka ei riittänyt paljon muuhun, ja koska piirtäminen oli tehtävänä selkeä, oli helppoa tavoitella tuottavuuden lisäystä sitä kautta. Tämän havaittiin kuitenkin kohta johtavan ongelmiin, koska piirtäminen on vain jäävuoren huippu tuotteen kuvaamisessa ja hyvin pieni osa suunnitteluprosessin kustannuksissa.

Nykyisin CAD-järjestelmillä pystytään tuotteiden monipuoliseen kuvaamiseen. Ongelmana on kuitenkin yhä suunnittelutietämyksen esittäminen ja mallintaminen. Ilman huomattavaa sovittamista monet CAD-työkalut eivät myöskään kykene yhdistämään tuotesuunnittelun eri osa-alueita. Näihin ongelmiin on viime vuosina pyritty löytämään ratkaisu tuotemallinnuksesta.

4.4 TUOTEMALLI

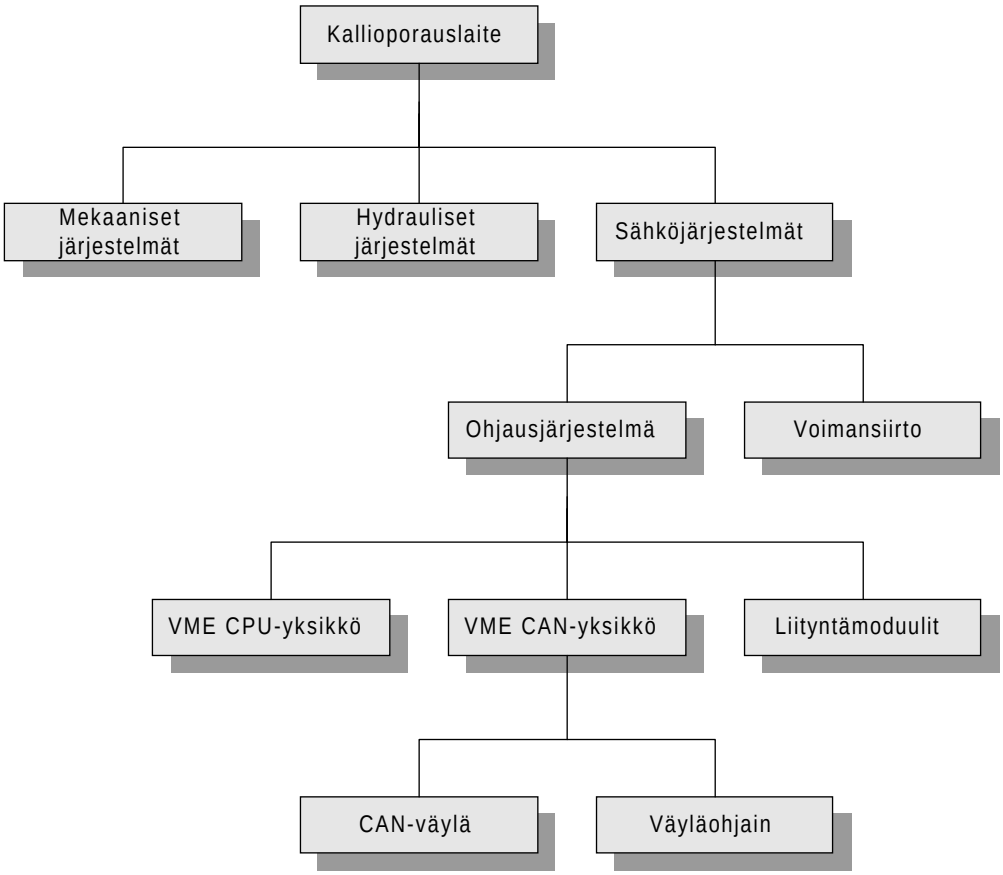
Tuotemalli on käsitteellinen malli tuotteen kuvaamiseen käytettävistä tiedoista ja niiden jäsennyksestä. Se on tapa kuvata tuotetta esittämällä esimerkiksi tuotteen sisältämät osat, miten ne liittyvät toisiinsa, mistä ne koostuvat ja mitä ominaisuuksia osilla on. Tuotemallin rakenne on yleensä hierarkkinen eli osakokonaisuudet koostuvat alemman tason osakokonaisuuksista, jotka taas koostuvat seuraavan tason osista, jne. [13]. Koostumuksen lisäksi tuotemallin tulee kuvata myös monia muita osien välisiä suhteita ja liityntöjä, jotka voivat olla myös

abstrakteja. Tuotemallit pohjautuvat yleensä erilaisiin informaatiomalleihin, joista esimerkiksi ER-malli (Entity-Relationship) [14] on paljon käytetty.

Hyvällä tuotemallilla tulisi olla seuraavanlaisia ominaisuuksia:

- Mallilla tulisi voida kuvata kaikki tuotetta koskeva tieto esitysmuodosta riippumatta.
- Ei tiedon päällekkäisyyttä eli kukin tieto on tallennettu vain kerran malliin.
- Malliin tulisi olla mahdollista lisätä ja muuttaa tietoa tuotteen koko elinkaaren aikana, esimerkiksi kunnonvalvontaan tai vikadiagnostiikkaan liittyen.
- Mallin tulisi olla riippumaton suunnitteluympäristön laitteistosta ja ohjelmistosta.

Tämän työn kohdejärjestelmänä käytetyn mekatronisen laitteen tuotemallin osa on esitetty alla olevassa kuvassa 5 ER-kaaviona, jossa oliot (entity) ovat tuotteen rakenteellisia osia ja suhteet (relation) “muodostuu” (part-of-) tyyppisiä.



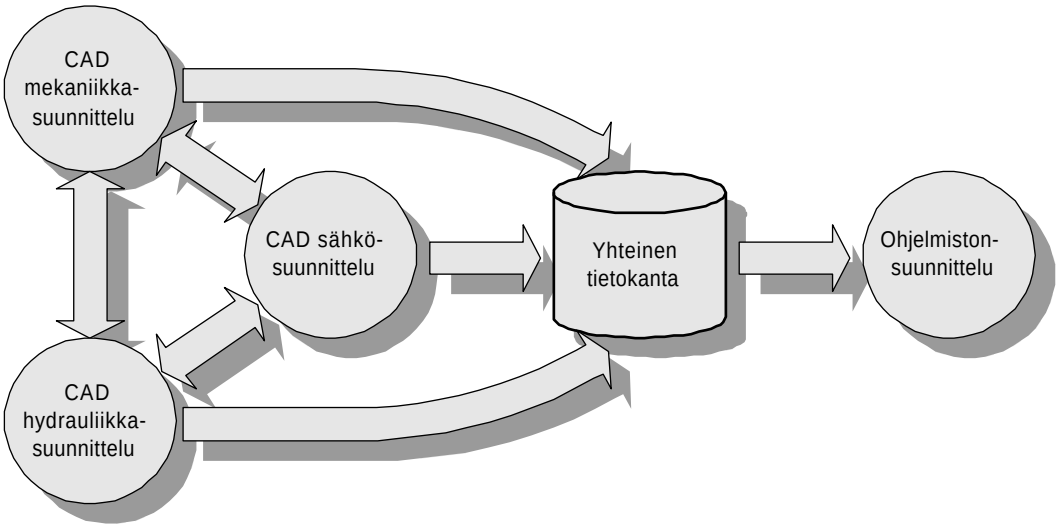
Kuva 5. Esimerkki mekatronisen laitteen tuotemallin osasta.

Mekatronisen laitteen rakennehierarkia voitaisiin lisäksi jakaa havainnollisuuden parantamiseksi tasoihin. Seuraavassa luetellaan mahdolliset tasot “ylhäältä alaspäin” eli suuremmasta kokonaisuudesta pienempään:

- Laitetaso: kuvaa laitteen rakenteen ja toiminnan kokonaisuutena
- Järjestelmätaso: kuvaa laitteen pääosa-alueet, esimerkiksi sähköjärjestelmä, mekaniikka, hydrauliikka.
- Osajärjestelmätaso: Järjestelmätaso voidaan jakaa toiminnallisiin osiin, esimerkiksi sähköjärjestelmä voidaan karkeasti jakaa osiin ohjausjärjestelmä ja voimansiirto.
- Osataso: Osat voivat olla konkreettisia fyysisiä osia, kuten esimerkiksi liityntämoduuli tai VME CPU -kortti.
- Komponenttitaso: Osatason oliot voidaan jakaa pienempiin komponentteihin. Komponentit ovat alkeisosia, jotka eivät enää jakaannu osiin. Esimerkiksi analogia-anturi kuuluu tälle tasolle.

4.5 TIETOKANNAN KÄYTTÖ TUOTETIEDON SIIRROSSA

Perinteisen CAD-suunnitteluprosessin tuotetiedon kuvaaminen on yksinkertaisinta toteuttaa käyttämällä CAD-suunnittelun osa-alueille yhteistä tietokantaa, kuva 6. Yhteiseen tietokantaan talletettua tuotetietoa voidaan käyttää moniin tarkoituksiin mekatronisen laitteen suunnitteluprosessin aikana. Yksi tärkeimmistä tietokannan käyttökohteista on toimia suunnittelutiedon varastona ja välittäjänä sulautettaviin ohjelmistosovelluksiin. Kun suunnittelutieto on koottuna yhteen paikkaan säännönmukaisina rakenteina, tiedon automaattinen muokkaus ohjelmisto-sovellusten osaksi tulee mahdolliseksi.



Kuva 6. Tietokannan käyttö CAD-suunnittelun ja ohjelmistosuunnittelun välisessä tiedonsiirrossa.

Varsinaisen tuotemallin luominen ei ole välttämätöntä, jos esitettävä tieto on rakenteeltaan yksinkertaista. Tietokannan käyttö ilman kehittyneempiä tuotemallinnusmenetelmiä ja standardisoituja suunnittelusääntöjä on käyttökelpoisimmillaan silloin, kun suunnitteluprosessia ei haluta ratkaisevasti muuttaa. Laajamittaisempi suunnittelutiedon ja -tietämyksen hyödyntäminen edellyttää perinteistä suunnitteluprosessia muuttavien oliopohjaisten tuotemallinnusmenetelmien [15] käyttöönottoa. Tuotemalli voidaan esittää oliopohjaisesti myös tavallisella relaatiotietokannalla, mutta tiedon määrän kasvaessa sen hallinta vaikeutuu huomattavasti. Kunnonvalvonnassa ja vikadiagnostiikassa tarvittavan tietämyksen kuvaaminen on mahdollista tietokannassa, jos kyseisen tietämyksen esittämisen ja tallentamisen säännöt määritellään erikseen suunnitteluprosessissa. Tällöin saattaa kuitenkin ilmetä ongelmia, mikäli suunnittelutyökalujen ominaisuudet ovat riittämättömät.

5 CAD-SÄHKÖSUUNNITTELUTIEDON HYÖDYNTÄMINEN

Työn sovelluskohteena on Tamrock Oy:n SOLO 1000 Sixty-kallioporauslaite. Tamrock suunnittelee, valmistaa ja markkinoi kaivosalan laitteistoa ja palveluita. Sen tuoteperhe ulottuu pienistä pintaporauslaitteista maanalaiseen suuren mittaluokan kaivaukseen tarkoitettuihin automaattisiin monipuomisiin porausjumboihin.

5.1 KOHDEJÄRJESTELMÄN KUVAUS

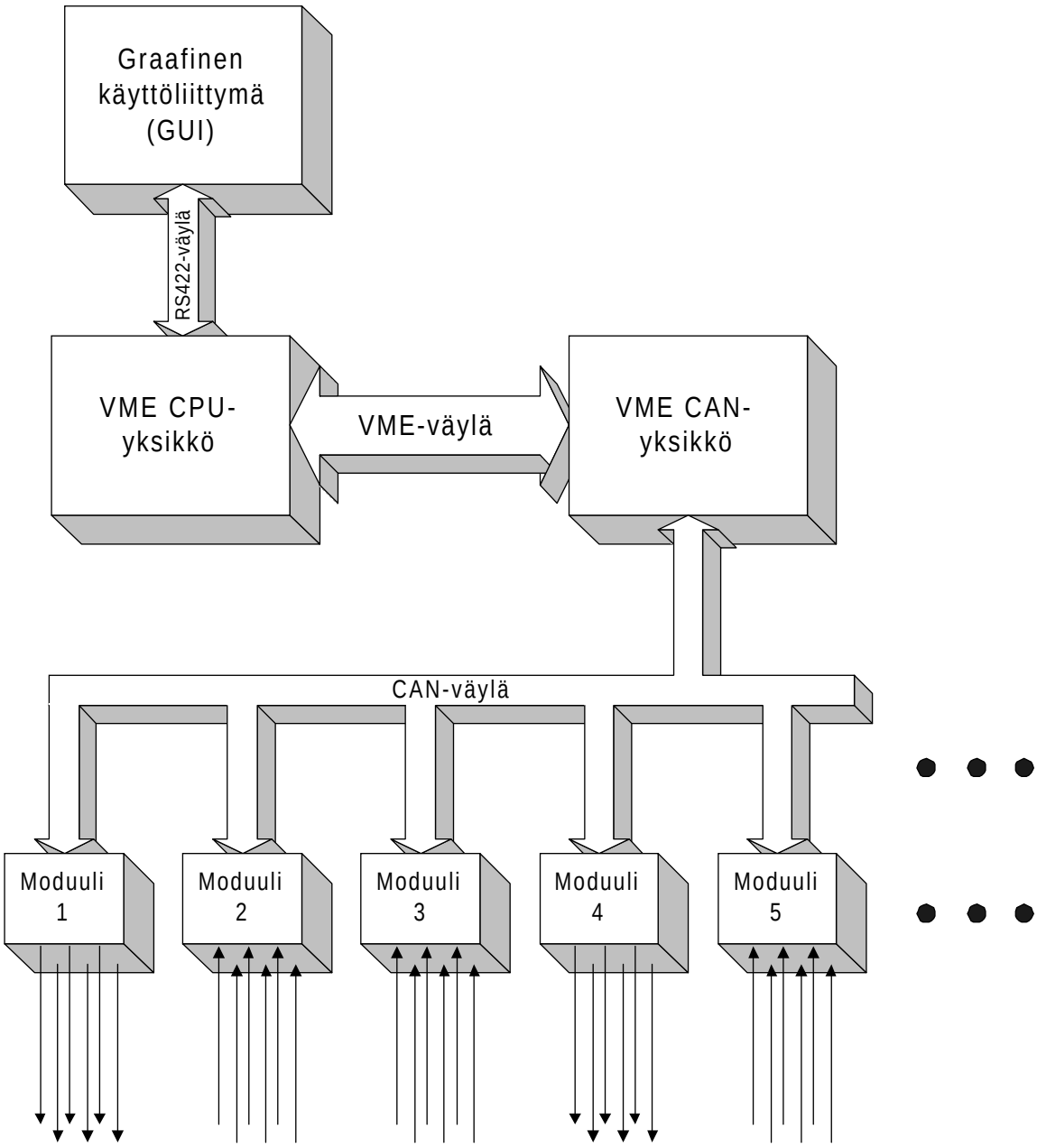
SOLO-sarjan kallioporauslaitteet edustavat Tamrock Oy:n uusinta tekniikkaa. Ne on tarkoitettu suuren mittakaavan maanalaiseen kaivaukseen ja rakentamiseen. Eniten laitteita käytetään malmin irrottamiseen kaivoksissa. Poraaminen voidaan suorittaa automatisoidusti käyttäjän valvoessa tehtävän suoritusta. Lounas- ja kahvitauot sekä vuoronvaihdot voivat tapahtua koneen työskennellessä. Automatisoidun poraamisen SOLO 1000 Sixty tekee lisäämällä poraustankoja reikään porauksen edetessä ja poistamalla ne reiän valmistuttua. Laite kykenee poraamaan 40 metriä pystysuoraan ylöspÄin ja 60 metriä sivuille ja alaspäin. Kuvassa 7 esitetään SOLO 1000 Sixty -kallioporauslaite käyttöympäristössään.

Kuva 7. SOLO 1000 Sixty -kallioporauslaite käyttöympäristössään.

Kallioporauslaite on vaativa mekatroninen laite, koska sen toimintaympäristö on normaalia ankarampi. Laite koostuu useista erilaisista hydraulisista, mekaanisista ja sähköisistä komponenteista sekä yli 20 anturista. Anturien määrää on pyritty vähentämään niiden kalleuden ja vikaantumisherkkyyden vuoksi. Ne tarjoavat tietoa laitteen toiminnasta ja ympäristöstä ohjausjärjestelmälle. Ohjausjärjestelmällä taas on kymmeniä lähtöjä, jotka ohjaavat laitteen toimintaa.

5.1.1 Ohjausjärjestelmä

SOLOn tietokoneohjattu toiminta on toteutettu modulaarisella TAS-ohjausjärjestelmällä (Tamrock Automation System), joka on yleiskäyttöinen ohjausjärjestelmä porauslaitteiden automatisointiin. Sen toiminta perustuu modulaariseen elektroniikkaan, joka mahdollistaa eri toimintoja omaavien liityntämoduulien hajauttamisen. Moduulit voidaan tällöin sijoittaa eri puolille kallioporauslaitetta lähelle ohjattavia toimilaitteita. Moduulien toiminnan ohjaus on kuitenkin keskitetty yhdelle prosessorille. Tämän keskusyksikön ja hajautettujen liityntämoduulien kommunikointi tapahtuu CAN-sarjaväylän välityksellä [16]. CAN-väylää ohjaa VME CAN -yksikkö, joka on kytketty suurempikapasiteettisen VME-väylän kautta keskusyksikköön. Keskusyksikkö kommunikoi tämän lisäksi RS422-sarjaväylän välityksellä graafisen käyttöliittymän (GUI) kanssa. Näitä yhteyksiä havainnollistaa parhaiten ohjausjärjestelmän rakennetta kuvaava periaatekaavio kuvassa 8. [17]



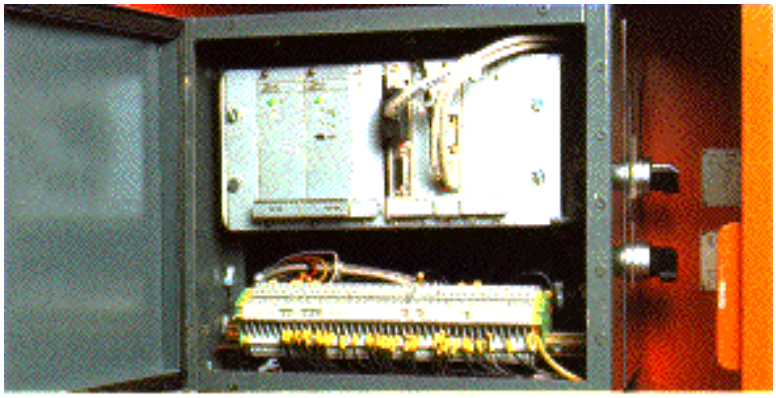
Kuva 8. Ohjausjärjestelmän rakenne.

Liityntämoduulit voivat toimia itsenäisinä ohjaimina tai VME CAN -yksikön ohjaamina. Kullakin moduulilla on omat yksilölliset tehtävänsä järjestelmän toiminnassa. Niiden avulla kerätään mittaustietoja toimilaitteiden tiloista ja antureiden arvoista. Keskusyksikön päättämät ohjauskäskyt toimilaitteille annetaan moduulien kautta. Moduulien lisäys ja poisto on mahdollista CAN-väylän ominaisuuksia käyttäen.

CAN-väylää voidaan kuvata kerrosrakenteella, johon kuuluu pohjalta alkaen fyysinen kerros, siirtoyhteyskerros, oliokerros ja sovelluskerros. Fyysisessä kerroksessa muodostetaan bittivirran esittämiseen tarvittavat signaalitasot sekä oikeat ajoitukset. Siirtoyhteyskerroksessa suoritetaan tarvittava bittivirran koodaus

käsitteellisempään muotoon. Siirtoyhteyskerros hoitaa myös virheiden ilmaisun ja raportoinnin, erilaiset kuittaukset ja viallisten solmujen rajaamisen. Oliokerros suorittaa vastaanotettujen viestien karsinnan ja lähetettävien viestien priorisoinnin. Jokaisella CAN-väylälle lähetetyllä viestillä on oma tunnistenumeronsa. Viestin lähettävällä tai vastaanottavalla asemalla ei ole varsinaista osoitetta, vaan vastaanottaja ja lähettäjä selvitetään viestin tunnisteen perusteella. Asema ottaa viestin vastaan, jos tunnistenumero vastaa sen käyttämää tietoa. Jokainen asema voi lähettää tai pyytää toista asemaa lähettämään tietoa. [16, 18]

Ohjausjärjestelmän VME CPU -yksikkö on koko järjestelmän keskusyksikkö, kuva 9. Se ohjaa systeemin funktioita prosessoimalla antureilta ja toimilaitteilta tulevaa tietoa ja lähettämällä sitä toimilaitteille. Prosessori suorittaa matemaattiset operaatiot mm. säätäjien ohjaamiseksi sekä tallentaa tarvittaessa tietoa ja järjestelmäparametreja [17]. Suoritettavat ohjelmat sulautetaan tallentamalla ne samalla kortilla sijaitsevaan PROM-muistiin. Keskusyksikkö hoitaa myös tiedonsiirron graafisen käyttöliittymän kanssa RS422-väylän kautta. Käyttö-järjestelmänä on OS9-reaaliaikakäyttöjärjestelmä [19].

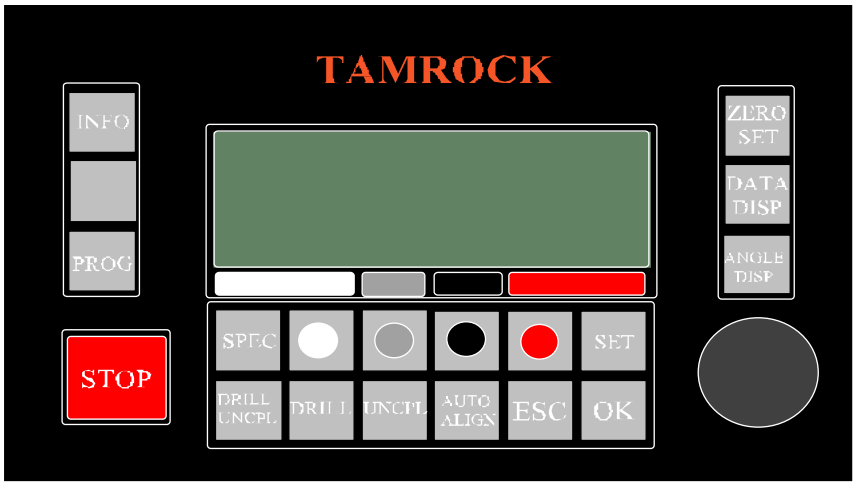


Kuva 9. Ohjausjärjestelmä on sijoitettu kallioporauslaitteen kylkeen.

SOLOa ohjataan ja sen toimintaa valvotaan ohjauspaneelin avulla, kuva 10. Kauko-ohjaus on mahdollista videokameran ja näytön avulla [17]. Graafinen käyttöliittymä sijaitsee ohjauspaneelissa ja se on kytketty kallioporauslaitteessa sijaitsevaan VME CPU -korttiin RS422-väylän välityksellä.

Kuva 10. Ohjauspaneeli, jossa keskellä graafinen käyttöliittymä.

Käyttöliittymässä on pieni graafinen näyttö toimilaitteiden tilojen, poraustiedon, asetuksien, hälytysten ja muiden tärkeiden viestien näyttämiseksi käyttäjälle. Näytön koko on 8 • 30 merkkiä ja 64 • 240 pikseliä. Käyttöliittymän näppäimiä ja säädintä voidaan käyttää käskyjen antamiseen CPU-kortilla ajettaville sovellusohjelmille. Lisäksi käyttöliittymä lukee signaaleja ohjauspaneelin kytkimiltä ja ohjaussauvoilta sekä ohjaa ilmaisinvaloja. Kaikki signaalit ohjauspaneelin ja kallioporauslaitteen välillä käyttävät RS422-väylää, paitsi hätäpysäytysnappi, jolle on omat johtimensa [17]. Kuvassa 11 esitetään graafinen käyttöliittymä pienennettynä noin puoleen normaalikoostaan.

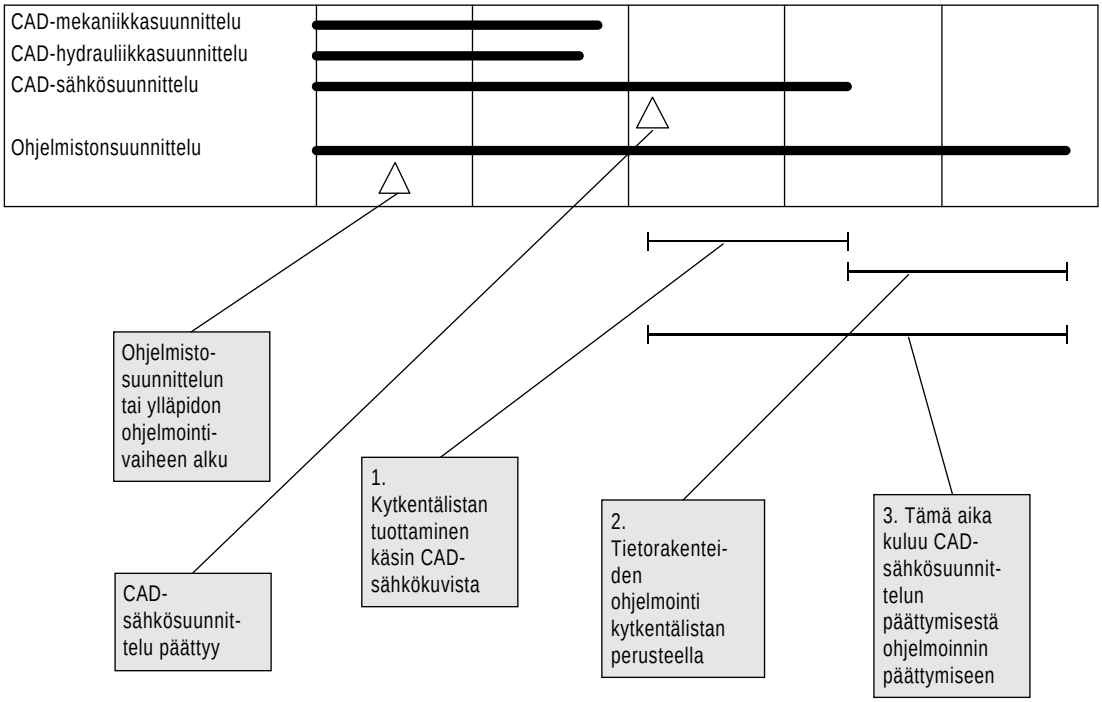


Kuva 11. Graafinen käyttöliittymä pienennettynä noin puoleen normaalikoostaan.

5.2 TEHTÄVÄN KUVAUS

Lähtökohtana oli tarve tehostaa suunnitteluprosessia CAD-suunnittelun ja ohjelmistosuunnittelun rajapinnassa. Tarkoituksena oli poistaa rajapinnasta kaksi aiemmin käsin tehtävää työvaihetta automatisoimalla tiedon siirto ja muokkaus

CAD-malleista sovellusohjelman tietorakenteiksi. Automatisoitavia välivaiheita olivat CAD-sähkösuunnittelussa liityntätiedon muokkaus kytkentälistoiksi ja ohjelmistosuunnittelussa sovellusohjelman tietorakenteiden ohjelmointi kytkentälistojen perusteella. Tavoitteena oli siis toteuttaa automaattinen suunnittelutiedon siirto sulautettuun järjestelmään yhdellä mekatronisen laitteen CAD-suunnittelun osa-alueella, sähkösuunnittelussa. Kuva 12 havainnollistaa suunnittelun ajallista etenemistä CAD- ja ohjelmistosuunnittelun rajapinnassa.

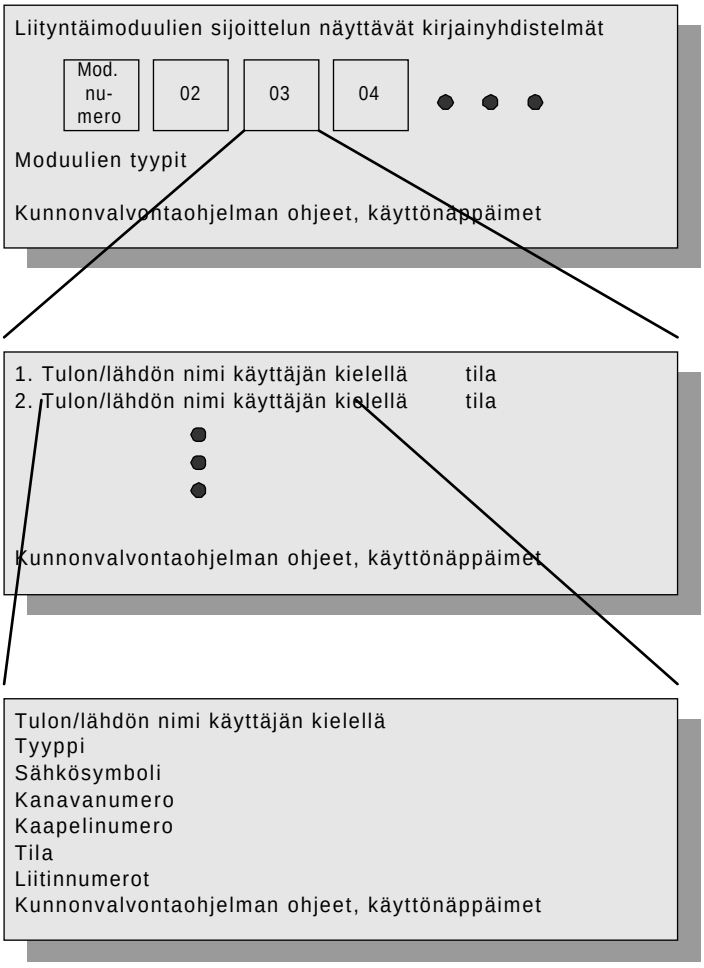


Kuva 12. Periaatteellinen ajoituskaavio CAD-suunnittelun ja ohjelmiston ohjelmointivaiheen välisestä ajallisesta yhteydestä.

Kuvasta nähdään suunnittelun eri osa-alueiden samanaikaisen etenemisen periaate Gantt-kaaviona. Ohjelmistosuunnittelulla tarkoitetaan tässä joko suunnittelun tai ylläpidon ohjelmointivaihetta. Ohjelmasovelluksen kiinteän rungon toteutus voi olla valmis jo ennen CAD-sähkösuunnittelun valmistumista, mutta sovelluksen käyttämät sähköjärjestelmän rakennetta kuvaavat tietorakenteet voidaan toteuttaa vasta kytkentälistan valmistuttua. Automatisoimalla kuvan tehtävät 1 ja 2 voidaan nopeuttaa tuntuvasti ohjelmointivaiheen päättymistä. Sellaisessa ylläpito-tilanteessa, jossa ohjelmasovelluksen kiinteä runko on muuttumaton ja tarvittavat tietorakenteet tuotetaan suoraan muutettujen CAD-mallien tiedon perusteella, ohjelmointia ei tarvita ollenkaan.

Tuotettuja tietorakenteita oli tarkoitus hyödyntää kohdejärjestelmään sulautettavassa reaaliaikaisessa kunnonvalvontaohjelmassa, joka piti suunnitella ja toteuttaa. Kunnonvalvontaohjelmalta vaadittiin selkeä ja yksinkertainen tiedon välittäminen käyttäjälle laitteen eri osien toiminnasta, antureiden arvoista ja sähköjärjestelmän rakenteesta reaaliajassa. Tiedot piti pystyä esittämään usealla

eri kielellä. Tavoitteena oli käyttää hyväksi CAD-sähkömalleista saatua tietoa liityntämoduulien sekä toimilaitteiden ja osajärjestelmien tulojen ja lähtöjen erityyppisistä ominaisuuksista. Sovelluksesta haluttiin hierarkkinen. Ensimmäisessä kuvassa näytettäisiin liityntämoduulien tyypit, sijainnit ja tilat. Seuraavaksi käyttäjä voisi tarkentaa tiettyyn moduuliin, josta hän pystyisi selailemaan kaikki moduulin tulot ja lähdöt sekä niiden tilat. Tämän jälkeen käyttäjällä olisi vielä mahdollisuus valita tietty tulo tai lähtö tarkempia tietoja halutessaan. Tiedot piti sovittaa kohdejärjestelmän graafisen käyttöliittymän näytölle. Kuvassa 13 esitetään käyttäjälle näkyvä tietojen esitystapa käyttöliittymän näytöllä. Lisäksi täytyi suunnitella helposti ymmärrettävä ja looginen kunnonvalvontaohjelman käyttöliittymä ohjeineen laitevalmistajan suunnitteluperiaatteiden mukaisesti.

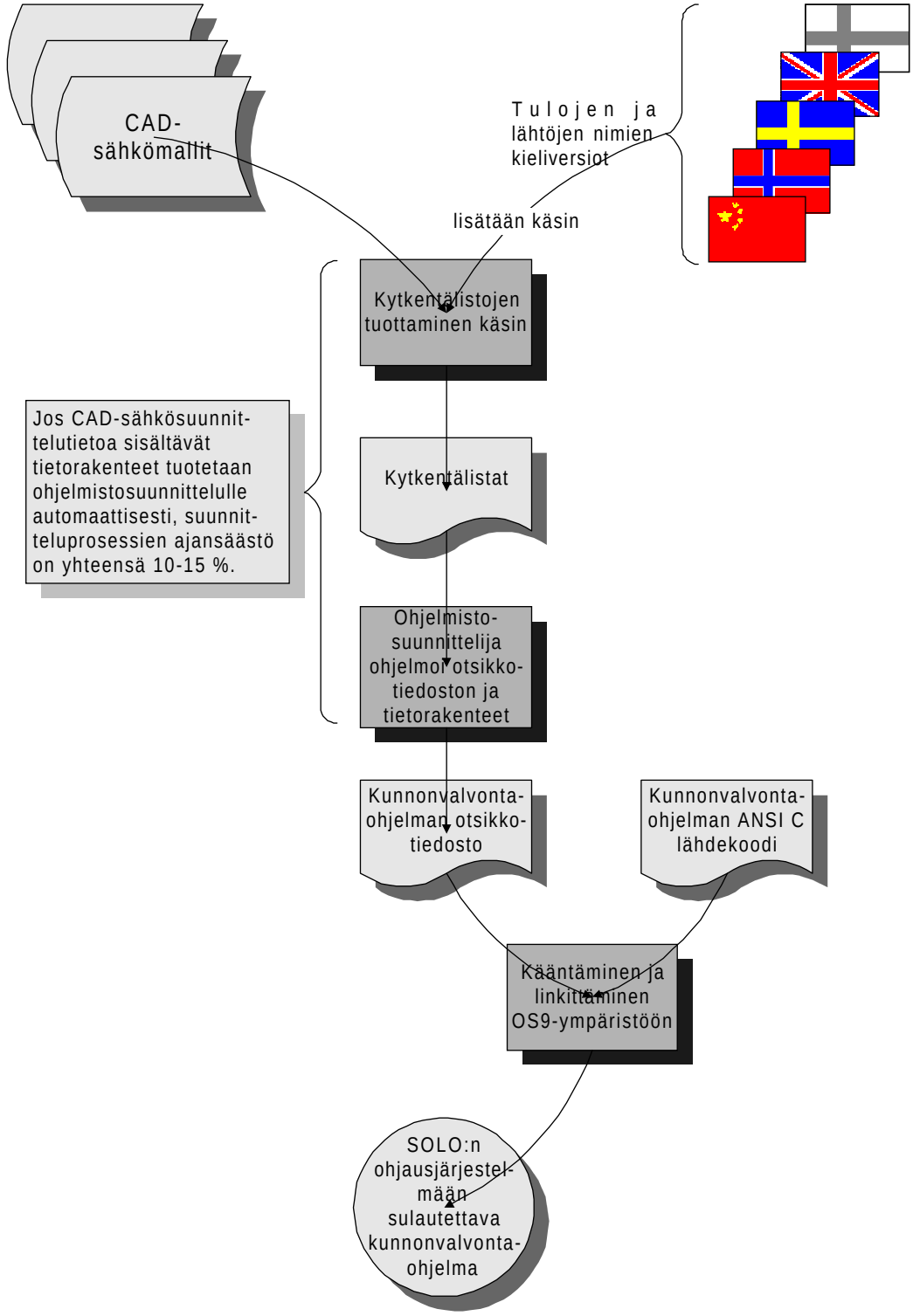


Kuva 13. Kunnonvalvontaohjelmalle määritellyt näytöt.

Kunnonvalvontaohjelman fyysinen sijainti on VME CPU -kortin PROM-muisti. Käyttäjä kommunikoi sovelluksen kanssa graafisen käyttöliittymän avulla. Tiedonvälitys moduuleilta sovellukselle tapahtuu CAN-väylän kautta VME CAN -yksikölle ja sieltä VME-väylän kautta CPU-kortille.

5.3 ALKUTILANNE

Alkuperäinen suunnitteluprosessin vuo esitetään kuvassa 14.

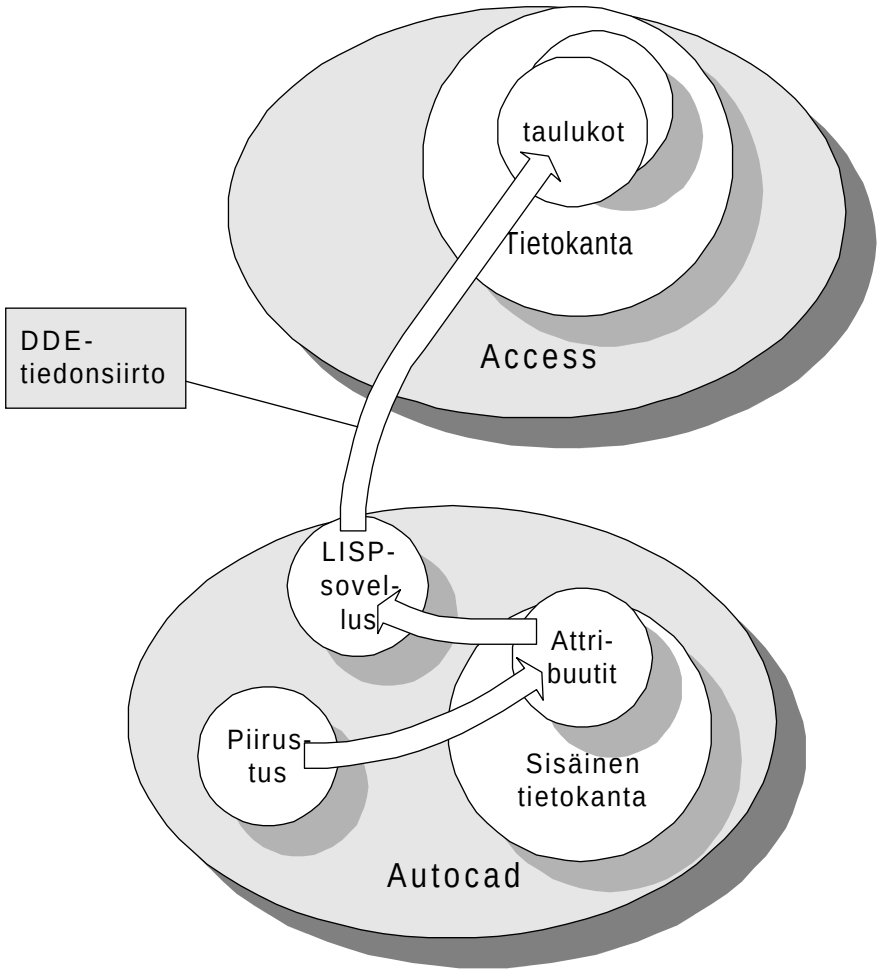


Kuva 14. Alkuperäinen suunnitteluprosessi.

Kuvasta nähdään, miten CAD-tiedon yhdistäminen kunnonvalvontaohjelmaan tapahtuu ilman suunnitteluprosessin parannuksia. Prosessissa on kaksi vaivalloista työvaihetta, jotka on mahdollista automatisoida. Ensimmäiseen niistä on syynä rajoitteita asettava CAD-järjestelmä. Käsin tapahtuva kytkentälistojen tuottaminen CAD-kuvien perusteella ei ole uusimmissa CAD-järjestelmissä tarpeen tietokantayhteyksien kehityttyä. Tällä on yhteys myös seuraavaan "ylimääräiseen" työvaiheeseen kyseisessä prosessissa: Tiedon ollessa paremmin organisoituna ja yhdessä paikassa sovellusten käyttämien tietorakenteiden automaattinen tuottaminen helpottuu huomattavasti.

5.4 CAD-SUUNNITTELUTIEDON TIETOKANTAYHTEYS

Uusimmissa CAD-suunnittelujärjestelmissä on useimmiten mahdollisuus tiedon suoraan siirtoon tietokantoihin. Mikäli tätä ominaisuutta ei ole, tarjotaan yleensä ainakin sovelluskehitysympäristö tietokantayhteyden toteuttamista varten. Esimerkkitapauksena CAD-järjestelmistä kokeiltiin Autodeskin AutoCAD 13:a Windows-ympäristössä [20]. AutoCAD-järjestelmässä piirustusolioiden attribuutit on tallennettu ohjelman omaan sisäiseen tietokantaan. Tällöin niiden käsittely on kuitenkin mallikohtaista ja rajoitettua eikä kovin joustavaa. Tässä tapauksessa haluttiin monen CAD-mallin tietyt attribuutit samaan paikkaan, missä niitä olisi helppo käsitellä ja niiden siirrettävyys olisi hyvä. Siksi valittiin käytettäväksi esimerkkitietokannaksi Microsoftin Access [21, 22]. AutoCAD tukee Windows-ympäristön DDE-tiedonsiirtoa, kuten myös Access. CAD-mallin attribuuttien siirto Access-tietokantaan saatiin toteutettua tekemällä sovellus AutoCADin AutoLISP-kielellä. Sovellus käynnistää tarvittaessa DDE:tä käyttäen Accessin, avaa tietokantatiedoston ja siirtää määrätyt attribuutit niille osoitettuihin taulukoihin (kuva 15).



Kuva 15. AutoCADin ja Accessin välisen DDE-tiedonsiirron periaate.

Esimerkissä tiedonsiirto tapahtui yhdestä CAD-mallista tiettyyn tietokantaan. Samaan tietokantaan voidaan kuitenkin lisätä attribuutteja useista muista kuvista käyttäen kyseistä DDE-sovellusta. Sovelluksessa määritetään myös se, mitkä attribuutit kuvasta siirretään. AutoCAD sisältää myös mahdollisuuden tehdä omia työkaluvalikoita ja lisätä niihin toimintoja. Tätä ominaisuutta käyttäen tehtiin valikkotoiminto, josta siirto tapahtui nappia painamalla.

Tällä esimerkillä saatiin osoitettua kuvan 14 ylimmän työvaiheen automatisointi. Kytkentätiedot saadaan siirrettyä helposti yhteen tietokantaan, missä niitä voidaan tarpeen mukaan muokata. Tietojen ollessa yhdessä paikassa myös tulojen ja lähtöjen nimeämiskäytännön yhdenmukaistaminen on paremmin aikaansaatavissa. Nimet käännettynä laitteen toimitusmaiden kielille voidaan lisätä samaan tietokantaan tai haluttaessa CAD-malleihin attribuuteiksi. Kytkentälistojen työläs tuottaminen käsin useiden CAD-mallien perusteella ei ole enää tarpeen.

5.5 TIEDON ESITYSTAVAT

CAD-mallien tiedot päätettiin sijoittaa Access-tietokantaan kahteen taulukkoon. Toisessa oli tietoa tuloista ja lähdöistä ja toisessa liityntämoduulien tiedot. Kuvassa 16 on esimerkki tietokannan taulukoista.

Symbol	Name, Finnish	Name, English	Name, Swedish	Module number
B1	Iskun paineanturi	Percussion pressure transducer	Tryckgivare för slaget	01
B3	Syötön paineanturi	Feed pressure transducer	Tryckgivare för matning	01
H614	Hätäpysäytetty	Emergency stopped	Nödstopp	00
P100	Iskun tuntimittari	Percussion hourmeter	Slag timräknare	00
SH82	Käynnissä merkkivalo	Aggregat in running	Aggregat på gång	00
Y10	Ilmahuuhtelu	Air flushing	Luftspolning	02
Y168	Maatukien liiketieto	Jacks moving by diesel	Landfästen rörelse med diesel	02
Y22	Puomin kallistus eteen	Boom tilting forwards	Lutning av bomens framåt	03
Y9	Vesihuuhtelu	Water flushing	Vattenspolning	02

Type	Channel number	Cable number	Connector numbers
AI	0	8	22 23 24
AI	1	9	25 26 27 28 29 30 31 32
DO	4	5	13 14 15
DO	2	7	19 20 21
DO	0	4	10 11 12
DI	1	2	4 5 6
DI	2	3	7 8 9
DO	0	6	16 17 18
DI	0	1	1 2 3

a) Esimerkki tietokannan liityntämoduulien tulojen ja lähtöjen tiedoista

Number	Type	Location
00	DIO8	R1
01	PWM4	R1
02	DIO8	R2
03	DIO8	R2
04	UNI	R3
05	DIO8	R3
06	DIO8	R4
07	DIO8	R4
08	DIO8	R5
09	UNI	R5

b) Esimerkki tietokannan liityntämoduulien tiedoista

Kuva 16. Esimerkki tietokantaan siirretystä CAD-sähkösuunnittelutiedosta.

Kunnonvalvontaohjelmalle tämä tieto täytyi saada muutettua ANSI-standardin mukaisiksi C-kielisiksi tietorakenteiksi. Tietorakenteiden muodossa päädyttiin samaan jaotteluun kuin tietokantaesityksessä.

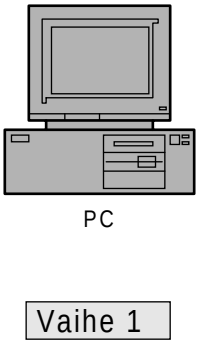
5.6 KUNNONVALVONTAOHJELMAN KUVAUS

Tavoitteena oli suunnitella ja toteuttaa reaaliaikainen SOLOn sähköjärjestelmän valvontaohjelma, jonka avulla käyttäjä saisi jatkuvasti tietoa laitteen senhetkisestä toiminnasta tai mahdollisista vikatilanteista. Ohjelma suunniteltiin käyttäen RT-SA/SD-menetelmää [5] ja Prosa-suunnitteluohjelmaa [23].

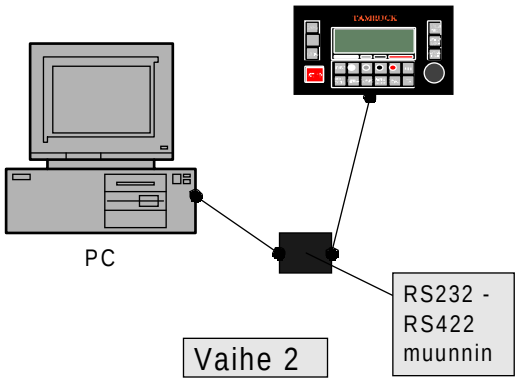
Käyttäjä käynnistää sovelluksen painamalla graafisen käyttöliittymän tiettyä näppäintä, jolloin tutkitaan moduulien tilat ja näytetään moduulikuva. Kuva sovittuu koko näytölle moduulien määrän mukaan. Tämän jälkeen palataan lukemaan käyttäjän mahdollisesti antama uusi käsky. Jatketaan käskyn ja tilojen tutkimista kunnes jompikumpi muuttuu. Jos tilat muuttuvat, näytetään heti uudet tilat. Käskyistä on tässä tilanteessa mahdollista antaa lopetus tai tietyn moduulin tulojen ja lähtöjen näyttämiskäsky valitsemalla jokin moduuli käyttöliittymän näppäimillä. Jälkimmäisessä tapauksessa tutkitaan valitun moduulin tulojen ja lähtöjen tilat ja näytetään ne sekä nimet halutulla kielellä. Tuloja ja lähtöjä voidaan selata sivu kerrallaan. Jälleen tutkitaan tiloja ja käyttäjän uutta käskyä, kunnes muutoksia tapahtuu. Käyttäjä voi palata edelliseen kuvaan tai valita tietyn tulon tai lähdön tarkempien tietojen katselua varten. Jälleen jäädään lukemaan tiloja ja seuraavaa käskyä, joka tässä tilanteessa voi olla ainoastaan paluu takaisin edelliseen näyttöön.

5.7 SIMULAATIOT

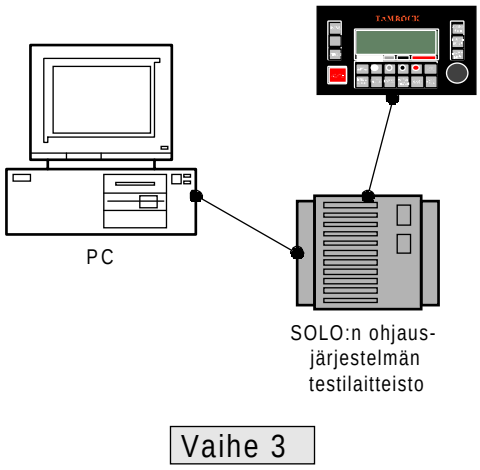
Suunnittelun jälkeen pyrittiin mallintamaan kunnonvalvontaohjelman toimintaa kahdella simulaatiolla lähestyen asteittain kohdejärjestelmän laitteistoa, kuva 17. Molemmat simulaatiot tehtiin Windows-ympäristöön Microsoftin Visual C++ -työkaluilla [24]. Simulaatioiden varsinaisten toiminnallisten osien ohjelmointikielenä käytettiin ANSI-standardin mukaista C-kieltä yhteensopivuuden vuoksi. Windows-käyttöliittymäosat tehtiin C++:lla Visual-ympäristön apuvälineitä käyttäen. Pyrkimyksenä oli tehdä simulaatiot mahdollisimman modulaarisiksi ja kohdejärjestelmän ympäristöön sulauttamista tukeviksi.



a) PC-simulaation laitteistojärjestely



b) Graafisen käyttöliittymän simulaatio



c) Sulautettavan kunnonvalvontaohjelman testausjärjestely

Kuva 17. Asteittainen siirtyminen kahden simulaation kautta kohdejärjestelmän laitteistoympäristöön. Kunnonvalvontaohjelman sijainti laitteistossa on merkitty harmaalla.

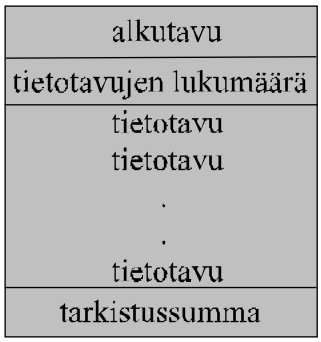
5.7.1 PC-simulaatio

Ensimmäinen simulaatio tehtiin kokonaan PC:lle käyttäen tulostukseen PC:n näyttöä ja käyttäjän käskyjen lähettämiseen näppäimistöä. Simulaatiosta tehtiin “Quickwin”-sovellus. Se ei ole varsinainen Windows-ohjelma, vaikka sisältääkin monia Windows-sovellukselle ominaisia piirteitä. Sovellus ei ole yhtä raskas, ja kehittäminen on suoraviivaisuudessaan DOS-ohjelmoinnin kaltaista. Käytettävissä olevien funktioiden määrä on kuitenkin ratkaisevasti pienempi kuin varsinaisessa Windows-sovelluksessa. Suurin osa käytettävistä funktioista on DOS-tyyppisiä, mutta edes kaikki ANSI-standardin mukaiset funktiot eivät toimi “Quickwin”-sovelluksessa.

Tällä simulaatiolla testattiin kunnonvalvontaohjelman rakenteen toimivuus sekä muokattiin tietorakenteita käytön kannalta parempaan muotoon. PC:n kuvaruudulle tehdyt mallinäytöt auttoivat hahmottamaan tietojen konkreettista esittämistapaa. Näppäimistön ja kuvaruudun käsittelyrutiinit pyrittiin sijoittamaan omiin aliohjelmiinsa siirrettävyyden parantamiseksi.

5.7.2 Graafisen käyttöliittymän simulaatio

Toisessa simulaatiossa oli tarkoitus ottaa käyttöön graafisen käyttöliittymän toiminnot tietojen esittämiseen ja käskyjen antamiseen. Käytännössä tehtiin PC-ohjelma, joka kommunikoi sarjaportin kautta graafisen käyttöliittymän kanssa. Sen ja PC:n sarjaportit täytyi sovittaa yhteen RS232-RS422-muuntimen avulla. Graafinen käyttöliittymä on noin 15 • 25 • 10 cm:n kokoinen laite, jolla on oma prosessorinsa saapuvien tietokehysten tulkitsemiseen ja sarjaliikenteen käsittelyyn. Kuten aiemmin mainittiin, siinä on näppäimistö sekä 8 • 30 merkin ja 64 • 240 pikselin graafinen nestekidenäyttö, joka tukee pikseli- ja viivagrafiikkaa. Lähetettävää ja vastaanotettavaa tietoa käyttöliittymä käsittelee kehyksinä, jotka koostuvat alkutavusta, kehyksen tavujen määrästä, tietotavuista ja tarkistussummasta, kuva 18.



Kuva 18. SOLOn graafisen käyttöliittymän tietokehys.

Graafisen käyttöliittymän simulaatiosta tehtiin Windows-sovellus. Edellisen simulaation kuvaruututulostustoiminnot muunnettiin käyttöliittymälle sarjaportin kautta lähetettäviksi tietokehyksiksi. Käyttäjän käskyt annettiin näppäilemällä käyttöliittymältä, joka muutti käskyt tietokehyksiksi ja lähetti ne sarjaporttiin, josta ne luettiin.

Visual C++ antaa sarjaliikenteen käsittelyyn kohtuullisen hyvät puitteet ylemmällä tasolla. Laitetasoa lähestyttäessä havaittiin kuitenkin puutteita funktiotarjonnassa.

Graafisen käyttöliittymän simulaation avulla saatiin aikaan toimiva tiedonsiirto kunnonvalvontaohjelman ja käyttöliittymän kanssa. Näytöt hioutuivat lopulliseen muotoonsa, kuten myös käyttäjän ohjelmalle antamat käskyt.

5.8 SULAUTETTAVA KUNNONVALVONTAOHJELMA

Toisessa vaiheessa oli tehty simulaatio, jossa kunnonvalvontaohjelma kommunikoi PC:ltä sarjaportin kautta graafisen käyttöliittymän kanssa. Nyt haluttiin muuttaa tätä simulaatiota siten, että se voitaisiin siirtää testilaitteistoon, jonka ympäristö vastaa SOLOn ohjausjärjestelmää. Laitteiston järjestely on esitetty kuvassa 17 c. Testilaite on kokoa 30 • 20 • 15 cm, ja sisältää VME CPU -kortin sekä kovalevyn. Tavoitteena oli saada kunnonvalvontaohjelma kommunikoimaan testilaitteelta graafisen käyttöliittymän kanssa sekä muuttaa se reaaliaikakäyttöjärjestelmään sopivaksi. Tämä edellytti myös kunnonvalvonta-ohjelman sovittamista SOLOn ohjausjärjestelmän muun ohjelmiston yhteyteen.

5.8.1 Käyttöjärjestelmä

SOLOn ohjausjärjestelmän käyttöjärjestelmänä [25] on Microwaren reaaliaika-käyttöjärjestelmä OS9 [19], joka tukee Motorolan 680x0 -prosessoriperhettä. OS9 tarjoaa enimmäkseen UNIX-tyyppisiä [26] ominaisuuksia, tiedostonhallinta, komen-totulkki, käyttöliittymä moniajomahdollisuuksineen, järjestelmäkutsut, putket, jne. ovat UNIXin kaltaisia. OS9 on kuitenkin paljon kompaktimpi käyttöjärjestelmä kuin UNIX ja muistinkäytöltään tehokas. Sen muita ominaisuuksia ovat muun muassa seuraavat:

- ytimen koko yleensä 25 kB
- monitasoinen priorisoitu keskeytyspalvelu
- priorisoitu round-robin skedulointi, prioriteetti käyttäjän muutettavissa
- monikäyttäjämahdollisuus
- mahdollisuus sijoittaa ydin PROM-muistiin
- keskeytyspohjainen I/O
- prosessien välinen viestinvälitys ja synkronointi.

5.8.2 Reaaliaikaisuusvaatimukset

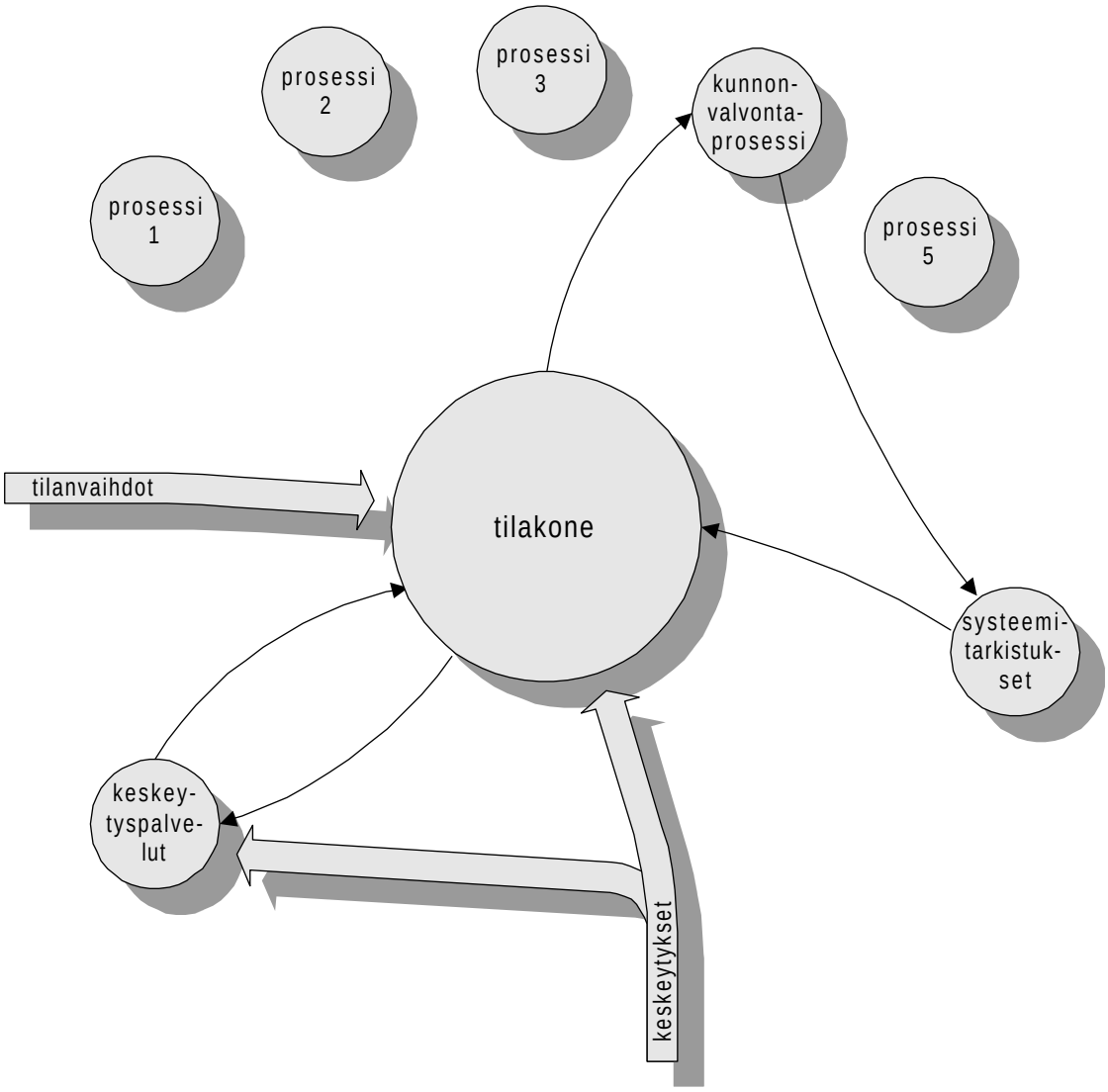
Ohjelmaa sanotaan yleisesti reaaliaikaiseksi, jos sen oikeellisuus riippuu oikean loogisen toiminnan lisäksi tulosten tuottamisen ajoituksesta. Ohjelman reaaliaikavaatimuksia sanotaan koviksi, jos aikarajojen ylittämisestä seuraa vikatilanne. Vastaavasti vaatimuksia kutsutaan pehmeiksi, jos aikarajojen väliaikainen ylittäminen on mahdollista, mutta siitä seuraa järjestelmän suorituskyvyn heikkeneminen. [27]

Ohjausjärjestelmän reaaliaikaisuus asetti kunnonvalvontasovellukselle uusia vaatimuksia. Tässä tapauksessa reaaliaikaisuusvaatimuksia voidaan kutsua pehmeiksi, koska aikarajoista lipsuminen ei aiheuta välittömästi virhetilannetta. Kunnonvalvontaohjelmalle oli asetettu tavoitteeksi mahdollisimman vähäinen prosessointiajan käyttö järjestelmän kuormituksen keventämiseksi. Ehdoton maksimiaika kunnonvalvontaohjelman käyttöön oli yhden sekunnin jakso kerrallaan. Tässä ajassa täytyi ehtiä suorittaa kaikki tarvittavat operaatiot mukaan lukien sarjaliikenne graafisen käyttöliittymän kanssa, joka oli kriittinen osa. Graafisen käyttöliittymän kanssa kommunikoivia prosesseja saattoi olla monia, jolloin sarjaliikenteestä voisi muodostua pullonkaula. Pahimmassa tapauksessa käyttäjä ei saisi tarvitsemaansa tärkeää tietoa ajoissa. Sarjaliikenteen nopeuttamiseksi käyttöliittymälle menevä tieto pitikin kerätä pidemmiksi kehyksiksi ja lähettää kerralla monien lyhyempien lähetysten sijasta.

5.8.3 Kunnonvalvontasovellus ohjausjärjestelmän osana

Kunnonvalvontaohjelman lopulliseksi fyysiseksi sijoituspaikaksi oli tarkoitus tulla VME CPU -kortin PROM-muisti. Kehitysvaiheessa sovellus siirrettiin PC:ltä testilaitteen kovalevylle, josta se ladattiin VME CPU -kortin RAM-muistiin.

Kunnonvalvontasovellusta täytyi muuttaa simulaatiosta siten, että siitä tuli tilakoneena toimivasta pääohjelmasta kutsuttava aliohjelma. Ohjaus ei saanut jäädä aliohjelmaan, vaan sen täytyi siirtyä sovelluksen läpi mahdollisimman nopeasti. Kunnonvalvontaohjelmalle varattiin tietty tila, johon siirryttyään tilakone kutsuisi sitä jatkuvasti, kunnes tila vaihdettaisiin. Aina ohjauksen poistuttua sovelluksesta suoritettiin erilaisia järjestelmätarkistuksia. Näiden tarkoituksena oli huomioida esimerkiksi kiireelliset käyttäjälle menevät hälytykset, jotka aiheuttaisivat väliaikaisen tilanvaihdon keskeytysrutiiniin. Hälytyksen loputtua palattaisiin takaisin entiseen tilaan. Luonnollisesti myös muita keskeytyksiä tapahtuu, mutta kaikki eivät välttämättä vaikuta sovelluksen toimintaan. Niitä ovat ainoastaan käyttäjälle menevät ylimääräiset ilmoitukset, joita ei ole mahdollista saada perille muuten kuin graafisen käyttöliittymän välityksellä. Kuvassa 19 nähdään kunnonvalvontasovellus osana muuta ohjelmistoa.



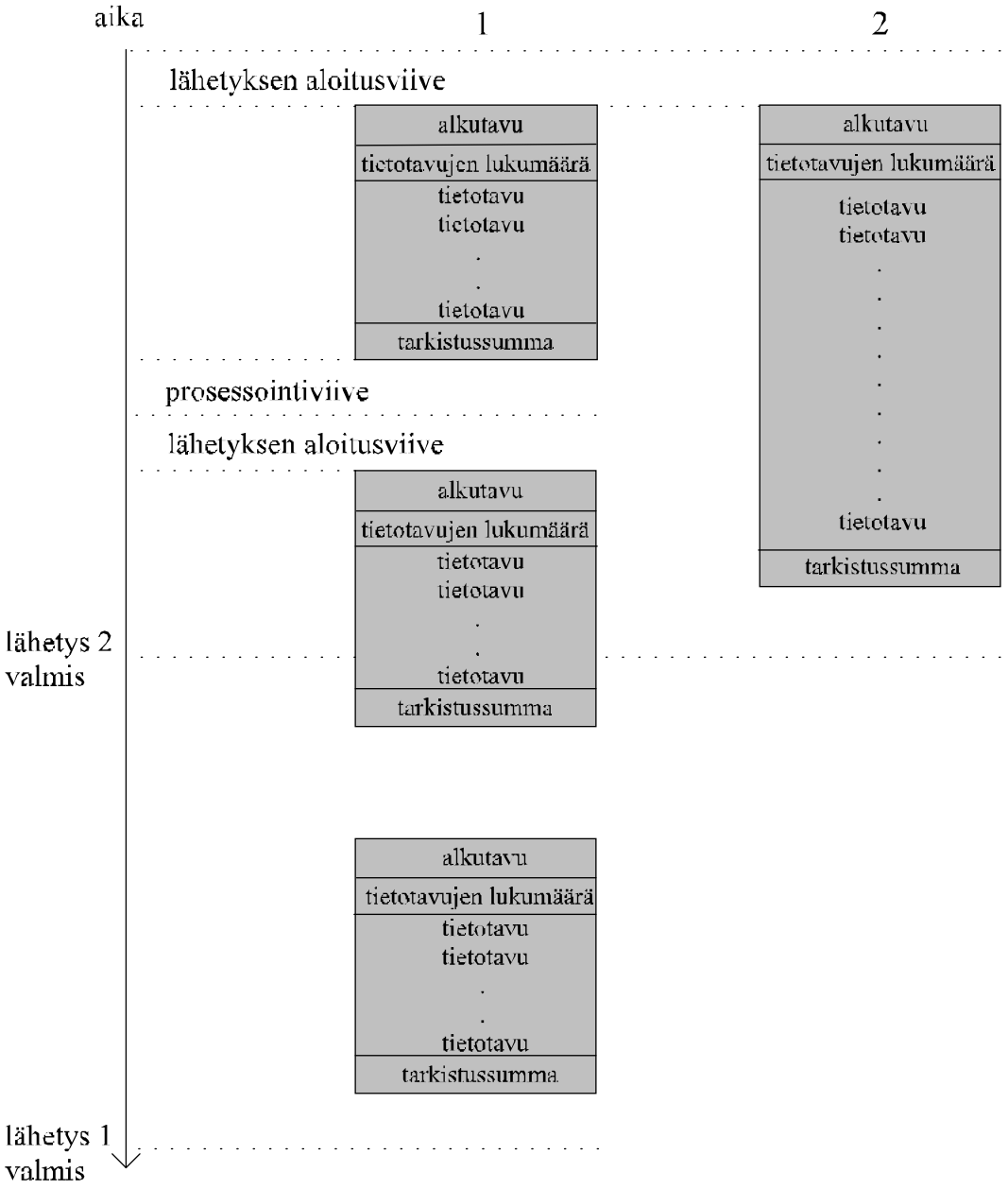
Kuva 19. Kunnonvalvontasovelluksen toiminta muun ohjelmiston osana.

5.8.4 Testaus

Sulautettavan kunnonvalvontasovelluksen ohjelmointi suoritettiin Visual C++:n editorilla. Lähdekoodi käännettiin DOS-ympäristössä ristikääntäjällä 680x0 -prosessorille ja linkitettiin ohjausjärjestelman ohjelmiston kanssa. Valmis ohjelmisto siirrettiin testilaitteen kovalevylle tiedonsiirto-ohjelmalla. Sovellusohjelman loogisen toiminnan oikeellisuus oli testattu jo simulaatioissa, joten jäljelle jäi tilojen lukemisen ja aikavasteiden tarkastelu.

Moduulien tilojen sekä tulojen ja lähtöjen tiloja voitiin mallintaa OS9-ympäristössä debuggerilla. Arvot voitiin syöttää PC:n näppäimistöltä VME CPU -kortille sarjaväylän kautta tiedonsiirto-ohjelman avulla. Tilojen vaihdon näkymisen aikavaste havaittiin riittäväksi, mutta graafisen käyttöliittymän näytön päivitys oli alussa hidasta. Ratkaisuksi tähän otettiin lähetettävän tiedon kerääminen

isommiksi kehyksiksi ennen lähetystä. Kuvasta 20 nähdään, miten kehyksen koko vaikuttaa lähetysnopeuteen.



Kuva 20. Kehyksen koon vaikutus lähetysnopeuteen.

Kehyksen osat kannatti kerätä suuremmaksi kokonaisuudeksi ennen lähetystä, koska tiedon hakuaika lähetysaikaan verrattuna oli merkityksettömän pieni. Periaatteessa lähettäminen nopeutuu sitä enemmän, mitä suurempi lähetettävä kehys on. Rajan kehyksen koolle asetti kuitenkin jo graafisen käyttöliittymän rajallinen vastaanottokyky. Liian suuret kehykset aiheuttivat tiedon katoamisen käyttöliittymän puskurin tultua täyteen.

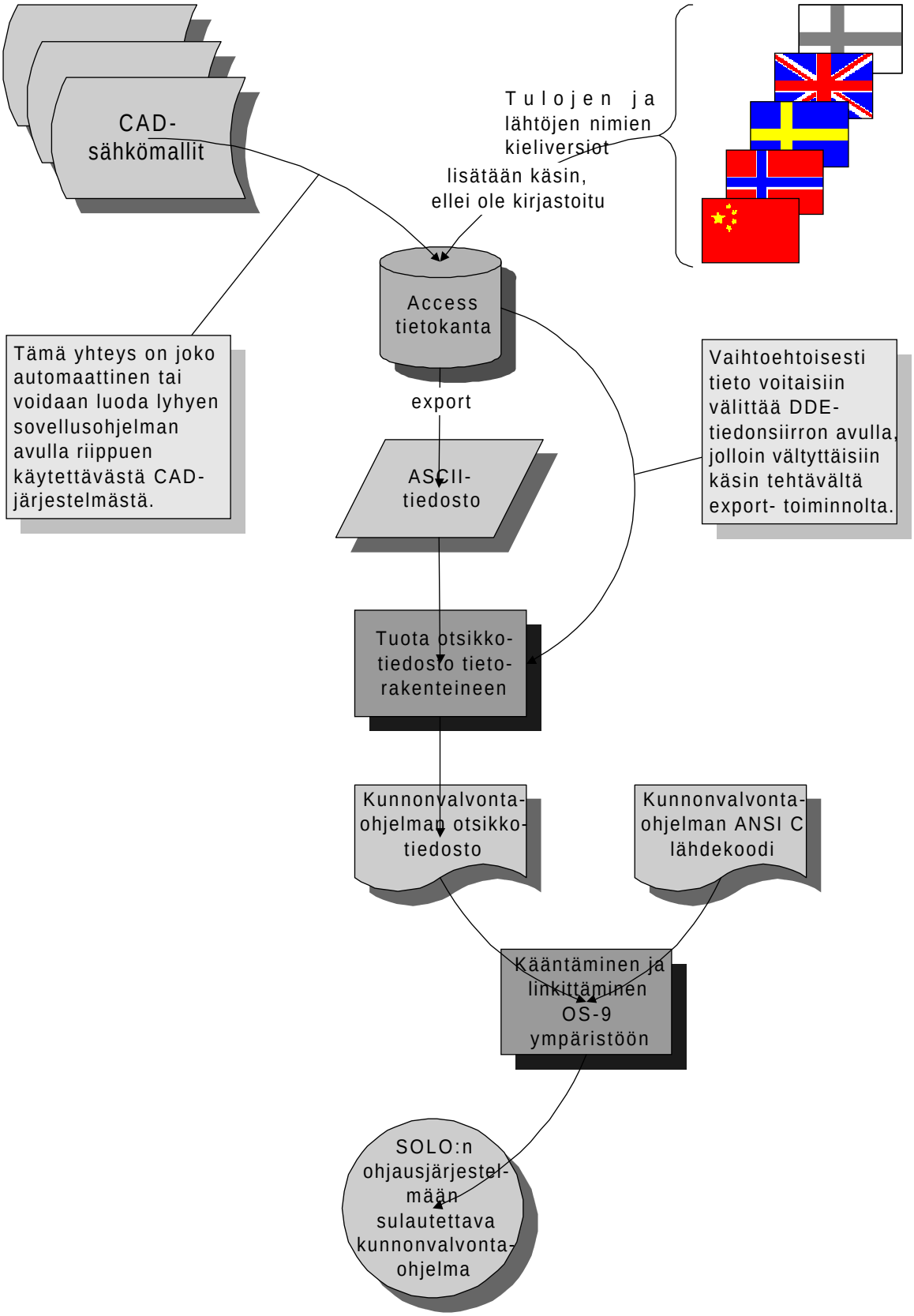
5.9 TIETORAKENTEIDEN AUTOMAATTINEN TUOTTAJA

CAD-kuvista tietokantaan siirretty tieto piti muuntaa kunnonvalvontaohjelmalle ANSI-standardin mukaisiksi C-kielisiksi tietorakenteiksi. Tietorakenteiden esitysmuodossa päädyttiin samaan jaotteluun kuin tietokantaesityksessä. Tietokantaesityksestä on esimerkki kuvassa 16. Tietorakenteiden tuottajaohjelma hyödyntää tietokannasta export-käskyllä tulostettuja ASCII-tiedostoja. Sovellus laskee tiedostoista joitakin kunnonvalvontaohjelman käyttämiä vakioita, kuten esimerkiksi moduulien lukumäärän ja kieliversioiden määrän. Luetut tiedot tulostetaan C-kielisiksi tietorakenteiksi tiedostoon, jota voidaan käyttää kunnonvalvontaohjelman otsikkotiedostona. Liitteessä 1 on esimerkki sovelluksen tuottamasta tiedostosta. Esimerkki on tuotettu käyttäen lähteenä kuvan 16 tietokantaa.

Automaattisesta tietorakenteiden tuottajasta tehtiin Windows-ympäristössä toimiva sovellus Microsoftin Visual C++-sovelluskehitystyökaluilla. Käyttöliittymää lukuun ottamatta sovelluksen ohjelmointiin käytettiin C-kieltä.

5.10 TULOKSET

Tavoitteiden toteutumista voidaan tarkastella vertaamalla alkuperäistä suunnitteluprosessia parannettuun prosessiin, kuva 21.



Kuva 21. Parannettu suunnitteluprosessin vuo esimerkkitapauksessa CAD-sähkösuunnittelun ja ohjelmistosuunnittelun rajapinnassa.

CAD-mallien ja tietokannan välinen tiedonsiirto saadaan aikaan CAD-järjestelmien tarjoamien sovelluskehitysympäristöjen avulla, tai vaihtoehtoisesti tietokantayhteys voi olla jo olemassa riippuen käytettävästä CAD-järjestelmästä. Esimerkki-tapauksessa toteutettiin AutoCADin ja Accessin välinen tiedonsiirto AutoLISP-kielisellä Windowsin DDE:tä hyödyntävällä sovelluksella. Tietokannasta tieto tulostettiin ASCII-tiedostoiksi jatkokäsittelyä varten. Tämä vaihe ei ole välttämätön, vaan Windows-ympäristössä toimittaessa voidaan haluttaessa käyttää DDE:tä tiedon siirtämiseen ilman välitiedostoja. Tiedon muokkaamiseen kunnonvalvontasovellukselle tehtiin tietorakenteiden tuottaja, joka ASCII-tiedostoja hyödyntäen tulostaa otsikkotiedoston. Tässä tiedostossa on sovelluksen tarvitsema tieto ANSI C -kielisinä tietorakenteina. Vaikka nämä rakenteet onkin tehty kunnonvalvontaohjelman toimintaa ajatellen, voidaan niitä käyttää myös muissa sovelluksissa.

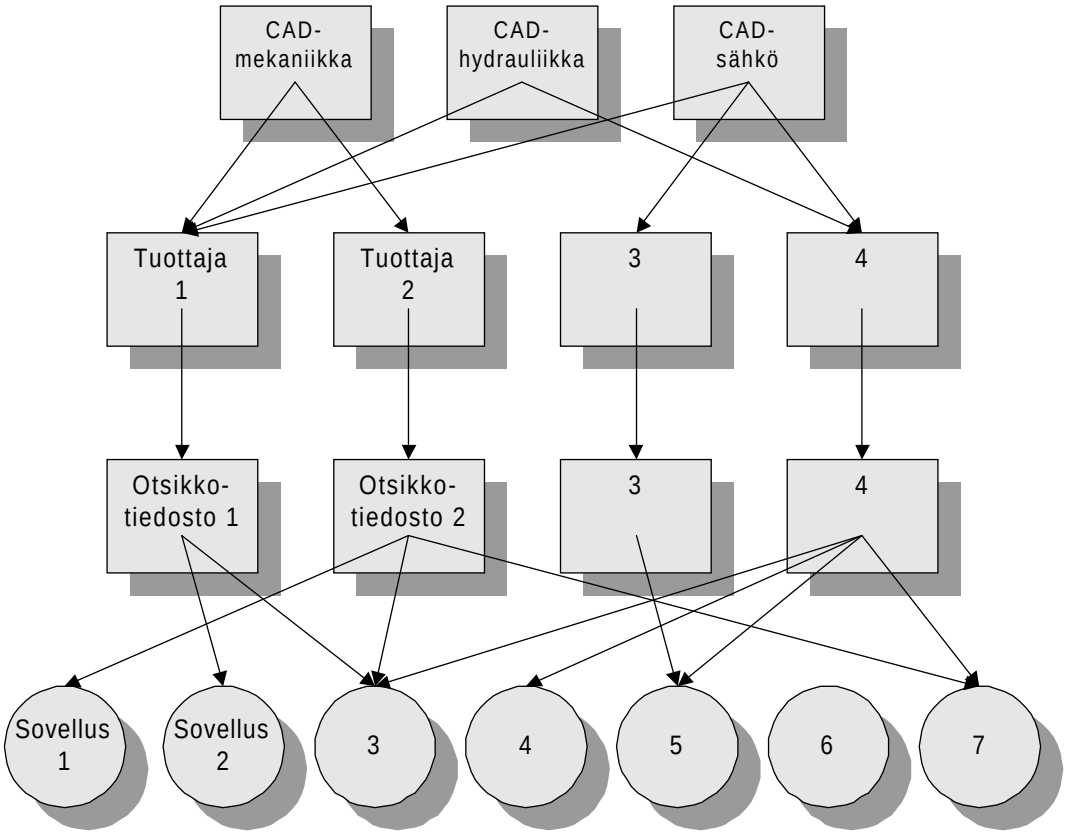
Kunnonvalvontaohjelman toiminnot saatiin määrittelyjen mukaisiksi. Sovellus testattiin kohdejärjestelmän ympäristöä vastaavassa testilaitteistossa ja havaittiin toimivaksi.

5.10.1 Hyödyt

Työssä esitetyllä menettelyllä tehostetaan suunnitteluprosessia CAD-sähkösuunnittelun ja ohjelmistosuunnittelun rajapinnassa. CAD-suunnittelu nopeutuu kytkentälistojen tuottamisen osalta ja ohjelmistosuunnittelu ohjelmointivaiheen osalta, kun tarvittavat tietorakenteet voidaan tuottaa automaattisesti suoraan CAD-malleista saadusta tiedosta. Tällä tavoin vähennetään ihmisen tekemiä työvaiheita ja samalla pienennetään myös virheiden määrää. Ylläpidettävyys paranee prosessin automatisoidulla osuudella huomattavasti, kun CAD-malleihin tehtyjä muutoksia ei enää tarvitse erikseen ohjelmoida sovelluksiin. Toimitusaikojen lyhentäminen on myös paremmin mahdollista suunnittelun osa-alueiden välisen tiedonsiirron kehittyessä.

Muita esitetyn menettelyn etuja ovat sen käyttöönoton nopeus ja sopivuus yrityksen CAD-suunnittelun kulkuun. Menetelmä on vaivaton ottaa käyttöön eikä aiheuta CAD-suunnitteluprosessin monimutkaistumista. Lisäksi menetelmä on järjestelmäriippumaton.

Saavutettuja tuloksia voidaan ajatella hyödynnettäviksi myös yleisemmin. Kun CAD-suunnittelu on standardoitua tai vakioitua esitystavoiltaan, tiedon automaattinen muokkaus ohjelmistosovellusten käyttöön on suoraviivaisempaa. Sovellusten käyttämä tieto voidaan esittää aina vakiomuodossa, jolloin ohjelmistosuunnittelijoiden ei tarvitse enää miettiä CAD-tietoa koskevien tietorakenteiden muotoja. Tällöin voidaan keskittyä ohjelmiston toiminnallisten osien suunnitteluun ja ottaa tarpeen mukaan käyttöön valmiiksi tuotetut rakennetietoa sisältävät osat. Kuvassa 22 havainnollistetaan tiedon siirtoa eri sovellusten käyttöön.



Kuva 22. Ohjelmistosuunnittelijat voisivat valita sovelluksiin tarvitsemansa tiedot automaattisesti tuotetuista otsikkotiedostojen tietorakenteista.

Mahdolliset lopputuotteen rakenteelliset muutokset voitaisiin päivittää vain CAD-malleihin, jonka jälkeen ajettaisiin tuottajaohjelmat ja käännettäisiin sovellusohjelmisto uudelleen. Tuotteen asiakaskohtainen sovittaminen tehostuisi tällä menettelyllä. Ohjelmiston ylläpitomahdollisuudet CAD-tiedon osalta olisivat tässä tilanteessa lähes ideaaliset.

5.10.2 Kehittämiskohteet

Esitetyllä menettelyllä on myös omat heikkoutensa. Sen avulla ei varsinaisesti tuoteta mitään korkean tason mallia järjestelmästä, vaan ryhmitellään ja muotoillaan tieto ohjelmistosuunnittelijan käyttöön. Tieto saatetaan CAD-järjestelmien esitysmuodosta formaaleiksi tietorakenteiksi, joiden ymmärtäminen ja käyttö on ohjelmistosuunnittelijan vastuulla. Tiedon esityksen täytyy tällöin olla erittäin hyvin ryhmiteltyä ja kuvaavasti nimettyä.

Lisäksi tulee ottaa huomioon tietorakenteiden automaattiset tuottajaohjelmat. Yksinkertaisesta rakenteestaan huolimatta niiden suunnittelu ja ohjelmointi vaatii

jonkin verran panostusta. Tätä seikkaa tärkeämmäksi noussee kuitenkin tuottajaohjelmien ylläpito CAD-suunnittelun esitysformaatin muuttuessa. Mikäli CAD-suunnittelun esitystapoja ei ole kunnolla yhdenmukaistettu, automaattisten tuottajaohjelmien ylläpito voi muodostaa ongelman. Lisäksi ongelmia seuraa varmasti, jos kaikki suunnittelijat eivät käytä samoja esitystapoja samoille asioille.

Yhdenmukaisuusongelmaan saadaan ratkaisu käyttämällä CAD-suunnittelussa standardoitua tuotemallinnusta. Esimerkiksi STEP-malli on tähän tarkoitukseen sopiva. Tällöinkin jonkinasteinen tuottajaohjelmien ylläpito on tarpeen, mikäli laitteen rakenteessa tapahtuu merkittäviä muutoksia. Tuotemallinnuksen käyttö takaa joka tapauksessa yhdenmukaiset suunnittelusäännöt ja -esitystavat sekä tuottaa havainnollisen mallin laitteesta.

6 JATKOKEHITYSMAHDOLLISUUDET

Jatkokehitykselle nähdään kaksi pääaluetta, jotka ovat kunnonvalvonta ja tuotemallinnus. CAD-suunnittelutiedon hyödyntämiseksi kunnonvalvonnassa käsitellään lyhyesti neljää eri lähitulevaisuuden kehitysmahdollisuutta. Toiselta pääalueelta tarkastellaan oliopohjaisia tuotemallinnusmenetelmiä, niiden hyödyntämismahdollisuuksia ja vaikutuksia suunnitteluun.

6.1 KUNNONVALVONTAOHJELMAN JATKOKEHITYSKOHTEET

Jatkokehityskohteiden pohdinnassa keskityttiin lähinnä työssä tehdyn kunnonvalvontaohjelman ominaisuuksien lisäämiseen. Seuraavassa on lista kehityskohteista lyhyine kuvauksineen:

1. Sähköjärjestelmän tulojen ja lähtöjen raja-arvojen syöttäminen voitaisiin tehdä CAD-suunnitteluvaiheessa. Menettely mahdollistaisi kunnonvalvontaohjelmiston reagoinnin sallittujen arvojen ylityksiin tai auttaisi käyttäjää näkemään tilanteen suhteessa normaaliarvoihin.

2. Tulojen ja lähtöjen tilojen fyysiset muistipaikat määritellään ohjelmistossa kahdella erillisellä tietorakenteella. Näiden tietorakenteiden automaattinen tuottaminen CAD-mallien tiedon perusteella tehostaisi lisää CAD- ja ohjelmistosuunnittelun välistä tiedonsiirtoa.

3. Hajautettujen liityntämoduulien fyysinen sijainti laitteessa voitaisiin kuvata CAD-suunnitteluvaiheessa. Tällä menettelyllä pyrittäisiin toteuttamaan moduulien sijainnin graafinen esittäminen kunnonvalvontaohjelmalla, mikä auttaisi käyttäjää vikatilanteessa välittömästi paikantamaan mahdollisen tiedonsiirtokatkoksen tai viallisen moduulin sijainnin laitteessa.

4. Eri osa-alueiden CAD-mallien osien rakenteellinen ja toiminnallinen liittäminen toisiinsa mahdollistaisi kunnonvalvonnassa toisiinsa liittyvien osien selailun helpottaen myös vikojen paikannusta osien välisten kytkentöjen perusteella. Osakohtaisen kunnonvalvontaa edistävän tietämyksen lisääminen CAD-malleihin olisi myös hyödyllistä.

6.1.1 Toteutusmahdollisuudet

CAD-suunnittelutiedon ja tietämyksen kuvaamiseen on CAD-suunnittelussa useita mahdollisuuksia. Jonkinasteinen tietämyksen esittäminen onnistuu ilman varsinaista tuotemallinnusmenettelyä käyttämällä hyväksi CAD-järjestelmän omia kuvausmenetelmiä ja attribuutteja. Seuraavassa käydään läpi edellä esitettyjen

jatkokehityskohteiden toteutusmahdollisuudet, edellytykset suunnittelulta ja vaikutukset suunnitteluprosessiin.

1. Tulojen ja lähtöjen raja-arvot voidaan lisätä CAD-malleihin vaivatta yksinkertaisina kuvaolioiden attribuutteina, joten toteutettavuus on hyvä. Tehtävä ei aiheuta mainittavaa lisätyötä CAD-suunnittelijoille eikä CAD-suunnitteluprosessin monimutkaistumista. Kyseinen tieto on myös helposti siirrettävissä sovelluksille tässä työssä käytetyllä menettelyllä.

2. Tulojen ja lähtöjen fyysiset muistipaikat määrittävien tietorakenteiden automaattinen tuottaminen edellyttää ohjelmistosuunnittelun muuttujien nimien generointia tulojen ja lähtöjen nimien kuvauksista. Voidaan esimerkiksi ottaa nimen kuvauksen kaksi ensimmäistä sanaa kokonaan ja seuraavista tietty määrä kirjaimia. Tämä taas edellyttää yhdenmukaista nimeämiskäytäntöä tuloille ja lähdöille CAD-suunnittelussa. Itse tietorakenteiden automaattinen tuottaminen on suoraviivaista toteuttaa nimeämisongelman ratkettua.

Tehtävän toteutus muuttaa vallitsevaa CAD-suunnitteluprosessia ainoastaan siten, että suunnittelijoiden täytyy nimetä tulot ja lähdöt yhteisten sääntöjen mukaan sen sijaan, että jokainen nimeäisi ne oman mielensä mukaan. Saavutettavaan ajansäästöön ja tiedonsiirron nopeutumiseen nähden vaatimukset suunnitteluprosessille ovat vähäiset.

3. Liityntämoduulien sijaintitietojen esittämistä CAD-malleissa pohdittaessa yhtenä keinona tuli esille sijainnin sanallinen kuvaaminen, mikä on pohjana graafiselle esitykselle kunnonvalvontasovelluksessa. Sanallisesti ilmoitettuna sijainti voidaan lisätä moduulin kuvaolion attribuutiksi, eikä laajempaa mallinnusta tällöin tarvita. Moduulien sijainnit voidaan ilmoittaa esimerkiksi seuraavasti: low_left_corner_of_, middle_right_center_of_, jne. Kunnonvalvonta-sovellus voi tulkita näitä attribuutteja ja esittää moduulin sijainnin graafisesti SOLOn kuvassa. CAD-suunnittelijoiden tulee sijainnin ilmoittamisessa käyttää yhdenmukaisia nimiä osista, joissa moduulit sijaitsevat.

Toinen vaihtoehto on käyttää tuotemallinnusta. Tällöin moduulien sijaintitiedot saadaan muun tuotetiedon ohella. Esimerkiksi STEP-malli on riittävän tarkka kuvaamaan laitteen rakenteen ja moduulien sijainnit laitteessa. Pelkästään tämän tehtävän toteuttamiseen tuotemallinnusta ei kuitenkaan välttämättä tarvita.

CAD-suunnitteluprosessiin sijaintitietojen esittäminen ei olennaisesti vaikuta. Suunnittelijoiden työmäärä lisääntyy yhden attribuutin kirjaamisen verran jokaista moduulia kohti.

4. Eri osa-alueiden CAD-mallien osien rakenteellinen ja toiminnallinen liittäminen toisiinsa on jatkokehityskohteista laajin ja vaativin. Piirustusolioiden attribuuttien hyödyntäminen ilman tuotemallinnusta on mahdollista, mutta tällöin tiedon määrä paisuu vaikeaksi hallita. Mekatronisessa laitteessa osien välisiä kytkentöjä on erittäin paljon, ja voidaan melkein sanoa, että kaikki vaikuttaa kaikkeen. Käytännössä tehtävä vaatii jonkin tuotemallinnusmenetelmän käyttöönottoa.

Tarkat säännöt tiedon ja tietämyksen kuvaamiseen sekä suunnittelun menettelytapojen määrittely ja yhdenmukaistaminen ovat tarpeen. STEP-standardissa nämä asiat on määritelty, ja Express-kieltä voidaan käyttää tuotetiedon kuvaamiseen. Toinen vaihtoehto tehtävän toteuttamiselle on oliopohjaisen tuotemallinnuksen soveltaminen tietokannan avulla. Tällöin vaaditaan huolellista tietokannan suunnittelua ja luonnollisesti suunnittelusääntöjen tarkkaa määrittelyä. Kolmas vaihtoehto on hyödyntää tarjolla olevia tietämyspohjaisia CAD-työkaluja.

Osien rakenteellisten ja toiminnallisten yhteyksien kuvaaminen sekä kunnonvalvonnan aputietämyksen lisääminen monimutkaistavat perinteistä CAD-suunnitteluprosessia huomattavasti lisäten CAD-suunnittelijoiden työmäärää. Tulevaisuudessa näiden tietojen sulauttaminen mekatronisen laitteen ohjausjärjestelmään tulee kuitenkin joka tapauksessa lähes välttämättömäksi. Saavutettavat hyödyt mm. kunnonvalvonnan ja vikadiagnostiikan kannalta ovat ratkaisevia. Oliopohjaiset tuotemallinnusmenetelmät tukevat myös osien kirjastointia, ja komponenttien uudelleenkäytöllä voidaan CAD-suunnittelijoiden työtä helpottaa ja työmäärää vähentää. Lisäksi laitteen elinkaaren aikana kertynyttä osakohtaista vikatietämystä voidaan tällöin uudellenkäyttää.

6.1.2 Yhteenveto jatkokehityskohteista

Kunnonvalvontaohjelman jatkokehityskohteiden vaikutuksista ja toteutettavuudesta esitetään yhteenveto kuvassa 23.

Ominaisuus	1. Tulojen ja lähtöjen raja-arvojen lisääminen CAD-malleihin	2. Fyysisten muistipaikkojen tietorakenteiden automaattinen tuottaminen	3. Moduulien sijaintitietojen esittäminen CAD-malleissa	4. Osien kytkentöjen ja tietämyksen esittäminen CAD-malleissa
Toteutettavuuden vaikeusaste	•	•••	•••	•••••
CAD- ja ohjelmistosuunnittelun rajapinnan tiedonsiirron tehostuminen	••	•••••	••	•••
Vaikutukset CAD-suunnittelijoiden työmäärään	•	•	•	••••
Tuotemallinnuksen käyttötarve	•	•	••	•••••
Hyödyt kunnonvalvonnalle	••	•	••	•••••
Hyödyt suunnitteluprosessille	•	•••	•	••••

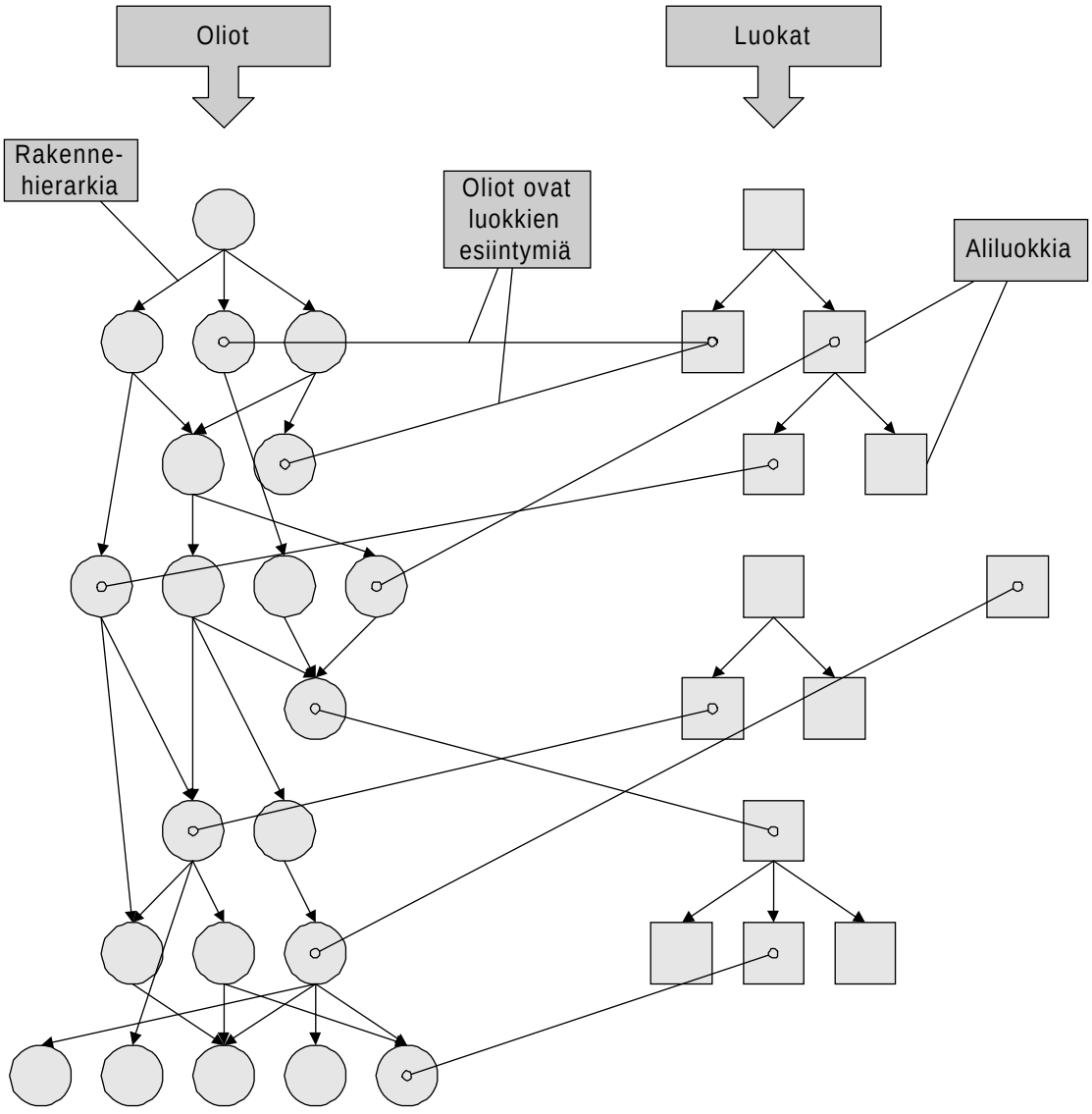
Kuva 23. Yhteenveto jatkokehityskohteiden toteutettavuudesta ja vaikutuksista (• = vähäinen, ••••• = huomattava).

6.2 TUOTEMALLINNUKSEN JATKOKEHITYSMAHDOLLISUUDET

Suunnittelutiedon kuvaaminen ja automaattinen siirto sulautettaviin ohjelmistosovelluksiin on mahdollista työssä esitetyllä parannetulla suunnitteluprosessilla. Rajan menetelmän toimivuudelle asettavat kuitenkin hyödynnettävän tiedon määrä ja monimutkaisuus. Niiden kasvaessa täytyy ottaa käyttöön standardisoituja suunnittelusääntöjä ja kehittyneempiä tuotemallinnusmenetelmiä.

6.2.1 Oliopohjainen lähestymistapa

Tuotemallinnuksessa käytetään yleensä oliopohjaisia kuvaustapoja. Olio voidaan tulkita tiettyä tuotetta, tuotteen osajärjestelmää, osaa tai komponenttia kuvaavaksi tietokoosteeksi. Olioilla on toisiinsa nähden erilaisia riippuvuuksia, kuten esimerkiksi koostumus, kytkeytyminen ja toiminnallinen yhteys. Yhteydet voivat olla myös abstrakteja. Luokat ja aliluokat ryhmittelevät oliot, joilla on yhteisiä attribuutteina ilmaistavia ominaisuuksia. Toisin sanoen luokissa määritetään olioiden tyypit. Olion voidaan sanoa olevan luokan esiintymä. Myös luokat ja aliluokat voivat olla olioita, mutta tässä yhteydessä käytetään selvyyden vuoksi pelkästään nimitystä luokka. Kuva 24 esittää oliopohjaisen tuotemallin periaatteen. [15, 28]



Kuva 24. Oliopohjaisen tuotemallinnuksen periaate.

Oliot ovat luokkien esiintymiä, joilla on luokissa määritellyt ominaisuudet. Luokkien aliluokilla on ylempien ominaisuudet sekä muuttuneet lisäominaisuudet. Esimerkiksi kytkinanturi ja analoginen anturi voisivat olla anturiluokan aliluokkia.

6.2.2 STEP

STEP (Standard for the Exhange of Product Model Data) [29] on kokoelma standardeja tuotetietojen kuvaamiseen ja tiedonsiirtoon. Standardin tarkoituksena on tarjota järjestelmäriippumaton tapa tuotetietojen kuvaamiseen tietokoneavusteisissa järjestelmissä tuotteiden koko elinkaaren ajalle. Standardin avulla pyritään eri organisaatioiden ja suunnittelun osa-alueiden välisen tiedonsiirron yhdenmukaistamiseen [29, 30]. STEP hyväksyttiin kansainväliseksi standardiksi maaliskuussa 1994 ja ANSI standardiksi helmikuussa 1995.

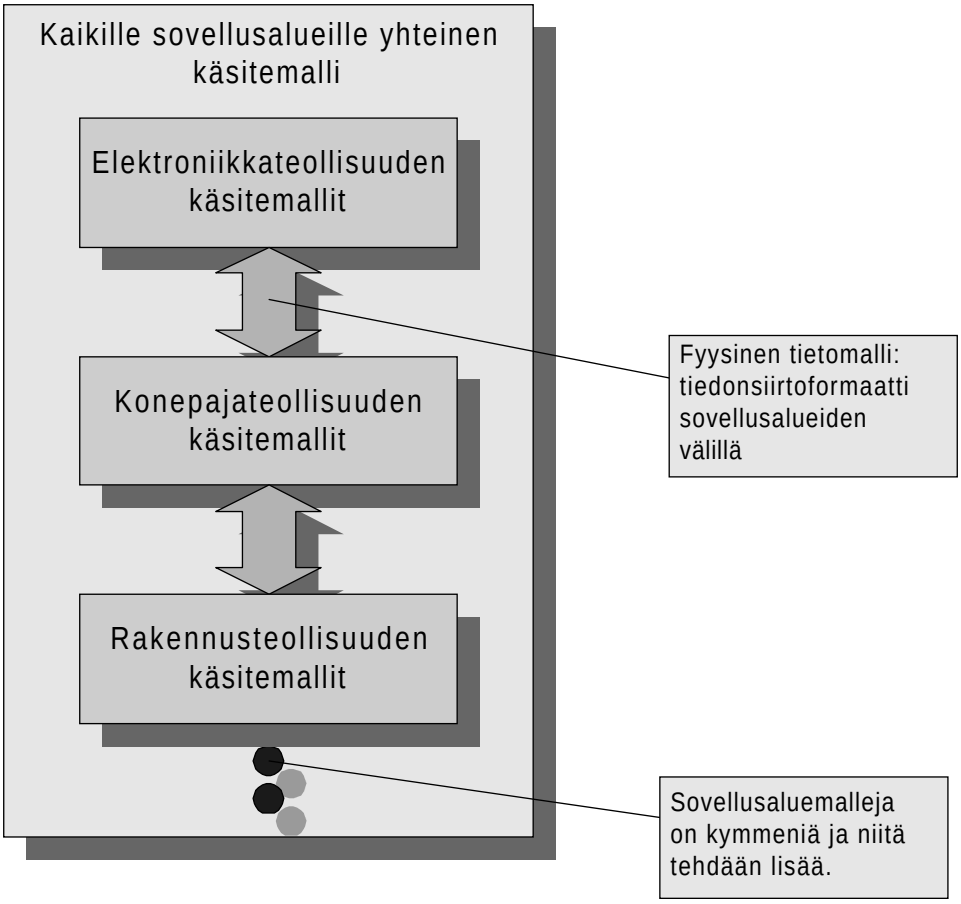
STEP-standardin mukainen tuotemalli on oliopohjainen. STEP-malli jakaantuu kolmeen pääosaan. Käsitetason malleina esitetään

- olioluokkien määritelmiä, jotka ovat yhteisiä kaikille tuotemallintamisen sovellusalueille
- olioluokkien määritelmiä, jotka ovat tyypillisiä jollekin sovellusalueelle, kuten esimerkiksi elektroniikkateollisuudelle, konepajateollisuudelle ja rakennusteollisuudelle.

Fyysisenä tietomallina esitetään

- siirtoformaatti, jonka avulla on mahdollista siirtää STEP-olioita järjestelmästä toiseen muuntamalla tiedot ensin siirtoformaattiin ja siirtoformaatista edelleen toisen järjestelmän käyttämään muotoon. [30]

Kuva 25 havainnollistaa STEP-mallin rakennetta.



Kuva 25. STEP-mallin ylemmän tason rakenteen periaate.

STEP-mallissa tuotteen kuvaamiseen käytetään Express-kieltä. Se on formaali, oliopohjainen tiedonkuvauskieli, joka on määritelty osana STEP-standardia. Tuotetieto kuvataan Express-kielessä olioluokkina, joihin liittyvät normaalit oliopohjaiset piirteet. Kielestä on olemassa myös luokkahierarkian esittämisessä havainnollisempi graafinen versio Express-G. [29]

Keskeinen Express-kielen käsite on olio. Se edustaa asiakokonaisuutta tai fyysistä kohdetta ja sen ominaisuudet kuvataan attribuuttien avulla. Attribuuteilla on nimi, joka kertoo attribuutin edustaman asian, sekä tietotyyppi. Express-kieli saa tärkeän osan ilmaisuvoimastaan antamalla attribuuteille laajan merkityssisällön. Attribuuttina voi perustietotyyppien, listojen ja matriisien lisäksi olla toinen olio. Sama piirre löytyy tietämystekniikassa käytetyistä kehyksistä. Tämä mahdollistaa joustavien sisäkkäisten luokkamäärittelyjen käytön. Express sallii myös attribuuteille määritellyt operaatiot funktioiden tai proseduurien muodossa. Lisäksi kieli sisältää mahdollisuuden käyttää ehtolauseita ja silmukoita, joiden avulla voidaan määritellä erilaisia sääntöjä. [29, 30]

Edellä mainituista tietolajeista oliot ja attribuutit saavat fyysisen toteutuksen tietokannoissa, säännöt ja operaatiot toteutetaan osana sovellusohjelmia tai tiedonhallintaohjelmistoja.

CAD-suunnittelun ja ohjelmistosuunnittelun välistä rajapintaa ajatellen STEP-malli tarjoaa mielenkiintoisia näkymiä. Jos tuotteen rakenne voidaan täysin kuvata Express-kielellä, tarvitaan vain kääntäjä, jolla Express voidaan muuntaa C++:lle tai C:lle, ja tuotetiedon siirto ohjelmistosuunnitteluun tehostuu ratkaisevasti.

Ratkaisu ei kuitenkaan ole aivan näin suoraviivainen mekatronisen laitteen suunnittelun kannalta. STEP-standardin sovelluskohtaiset mallit eivät välttämättä kata kaikkia osa-alueita mekatronisen laitteen suunnittelussa. STEP-standardi sisältää vain sellaisia olioluokkia, joita useat eri sovellukset voivat hyödyntää. Sovelluskohtaisessa käsitemallissa määritellään tietyn osa-alueen omat olioluokat, sekä mitä yleisiä olioluokkia sovellus tarvitsee. Käännösohjelmat STEP-standardissa toteutetaan siten, että ne täyttävät aina jonkin tietyn sovellusalueen vaatimukset. Mekatronisen laitteen CAD-suunnittelun ja ohjelmistosuunnittelun välisessä tiedonsiirrossa tämä tarkoittaa käytännössä sitä, että kun ohjelmistosuunnittelussa tarvitaan tuotetietoa suunnittelun useilta eri osa-alueilta, niille täytyy olla erilliset kääntäjät. Mahdollisiin muutostarpeisiin standardi pystyy vastaamaan hitaasti, jos ollenkaan. Tällöin joudutaan käytössä olevaa järjestelmää räätälöimään, mikä saattaa jälleen johtaa tiedonsiirtovaikeuksiin.

6.2.3 Tietämyspohjaiset CAD-järjestelmät

Viime vuosina on kehitetty myös suunnitteluympäristöjä, joissa CAD-järjestelmän yhteyteen on liitetty tietämyspohjaisia työkaluja tuotemallinnuksen mahdollistamiseksi. Esimerkiksi ICAD, Design++ ja Concept Modeller ovat tällaisia ympäristöjä. Näissä järjestelmissä käytetään tuotemallin kuvaamiseksi oliomenetelmiä. Tietämyksen esittäminen tapahtuu kehysten avulla [10, 11]. Perinteiseen relaatiotietokantaesitykseen verrattuna kehykset voivat sisältää monimuotoisempaa tietoa ja sääntöjä. Suunnittelusäännöt tekevät kokemusperäisen tietämyksen esittämisen ja hyödyntämisen helpommaksi ja järjestelmällisemmäksi. Kehysten käyttö mahdollistaa tietojen periytymisen ja sitä kautta attribuuttien dynaamisen hallinnan. Lisäksi olion ominaisuuksien kuvaamiseen ja käyttäytymiseen liittyvä tieto voidaan sisällyttää samaan tietorakenteeseen. Relaatiotietokannat ovat tehokkaita tiedon hallinnassa ja käsittelyssä, kehykset puolestaan ovat parempia monimutkaisissa päättelyprosesseissa. [31]

Suunnitteluprosessin kannalta CAD-järjestelmien kehityksen seurauksena on niiden käyttöalueen laajeneminen. Tietämyspohjaiset työkalut antavat mahdollisuuden tietokonetuettuun suunnitteluun koko suunnitteluprosessin aikana vaatimusmäärittelystä lopulliseen suunnitteluun saakka. Tietämyspohjainen CAD tukee myös moniteknologisten tuotteiden rinnakkaista insinöörityötä. [31]
Ongelmana tietämyspohjaisten CAD-järjestelmien käyttöönotossa on niiden aiheuttamat perusteelliset muutokset suunnittelutyöhön. Kokeneiden, toisenlaisiin menettelytapoihin tottuneiden suunnittelijoiden on vaikea omaksua kokonaan uutta suunnittelukulttuuria ja uusia sääntöjä. Tästä huolimatta CAD-järjestelmien

kehitys suuntautuu tulevaisuudessa yhä enemmän tietämyspohjaisiin, vahvasti integroituihin suunnitteluympäristöihin.

6.2.4 Oliomallin kytkeminen CAD-järjestelmään

VTT Rakennustekniikan OOCAD-projektissa [32] (Object Oriented Computer Aided Design) luotiin tuotemallinnussovellus hyödyntämällä CAD-järjestelmän tarjoamia tietorakenteita ja räätälöintimahdollisuuksia. Projektissa tehtiin prototyyppisovellus AutoCAD-järjestelmään. Sovelluksessa tuotemallin tallennuspaikkana oli AutoCADin oma kuvatietokanta, ja eri järjestelmien väliseen tiedonsiirtoon käytettiin projektissa kehitettyä OXF-tiedonsiirtoformaattia (Object Exchange File). [32]

OOCAD-tuotetietomalli on nimensä mukaisesti oliopohjainen. Mallissa tuotteen ominaisuudet kuvataan attribuutteina eli määreinä, jotka kootaan yhteen ominaisuus- eli attribuuttijoukoiksi. Attribuuttijoukot liitetään luokkiin, joiden ominaisuuksia niillä kuvataan. Esiintymäolioissa luokkien ominaisuusjoukkojen määreet saavat arvot. Lisäksi olioille voidaan antaa esiintymäkohtaisia määritteitä. Käyttäjä voi laatia ominaisuusjoukkoja haluamansa määrän haluamallaan tavalla, mutta käytännössä ne pitäisi standardoida esimerkiksi attribuuttikirjastoiksi. Tällöin sovellusohjelmat tietävät, mistä attribuutista kulloinkin on kyse, eikä määritteiden tulkinta jää vain käyttäjälle. [32]

Projektin tuotemalli toteutettiin AutoCAD-järjestelmän avulla. Yleinen CAD-järjestelmien piirre on se, että useita graafisia objekteja voidaan koota yhdeksi kokonaisuutena käsiteltäväksi koosteeksi eli symboliksi, jota AutoCADissä kutsutaan lohkoksi. Lohkoon voidaan liittää lisätietoa attribuuttien muodossa. Lohko tallennetaan määrittelynsä jälkeen kuvatietokannan sisäiseen symbolikirjastoon tai massamuistiin, mistä se voidaan lisätä piirustukseen. Kuvaan lisätty lohko voi sisältää myös muiden lohkojen esiintymiä, jolloin voidaan luoda hierarkkinen koostumusrakenne. Havaitaan, että lohko vastaa suoraan tuotetietomallin luokkaa: Kukin lohko määritellään vain kerran, sillä voi olla useita esiintymiä ja se voi koostua muiden lohkojen esiintymistä. Esiintymäolio on vastaavasti lohkon lisäys kuvaan. Lohkoja käytetään hyväksi myös ominaisuusjoukoissa, jotka ovat kooste attribuuttimääritelmistä, joilla ei ole arvoa. Lohkojen käytön hankaluutena on niiden myöhemmin tapahtuvan muokkauksen vaikeus. [32]

Kuvattu tuotemallisovellus vaatii toimivuuden saavuttamiseksi suhteellisen runsaasti CAD-järjestelmän räätälöintiä. Lisäksi se ei toimi kaikissa järjestelmissä, mikäli apusovellusten tekeminen ei ole yhtä hyvin järjestetty kuin AutoCADissä. Tuotemalli on kuitenkin tällä tavoin helposti toteutettavissa oliopohjaisena.
Tämän työn kannalta tärkeässä asiassa eli mekatronisen laitteen CAD-suunnittelun ja ohjelmistosuunnittelun rajapinnan tiedonsiirron tehostamisessa esitetty menetelmä ei ole paras mahdollinen. Tämä johtuu ennen muuta tietojen hajanaisuudesta. Ongelmia syntyy, jos vaaditaan monien eri suunnittelumallien

yhdistämistä, kuten mekatronisen laitteen suunnittelussa. Siirtotiedostot eivät ole tähän tarkoitukseen erityisen tehokas tapa.

6.2.5 Yhteenveto tuotemallinnusmahdollisuuksista

Tuotemallit ovat oliopohjaisia, koska se on luonnollinen tapa kuvata tuotteen jakaantumista osiin. Yhtä lukuun ottamatta esitellyissä mallinnusmenetelmissä käytetään tuotteen kuvaamiseen olioita. Yhteistä tietokantaa käytettäessä ei välttämättä luoda varsinaista mallia tuotteesta. Kuvassa 26 on yhteenveto tuotetiedon kuvaamisessa käytettävien menetelmien ominaisuuksista ja vaikutuksista suunnitteluun.

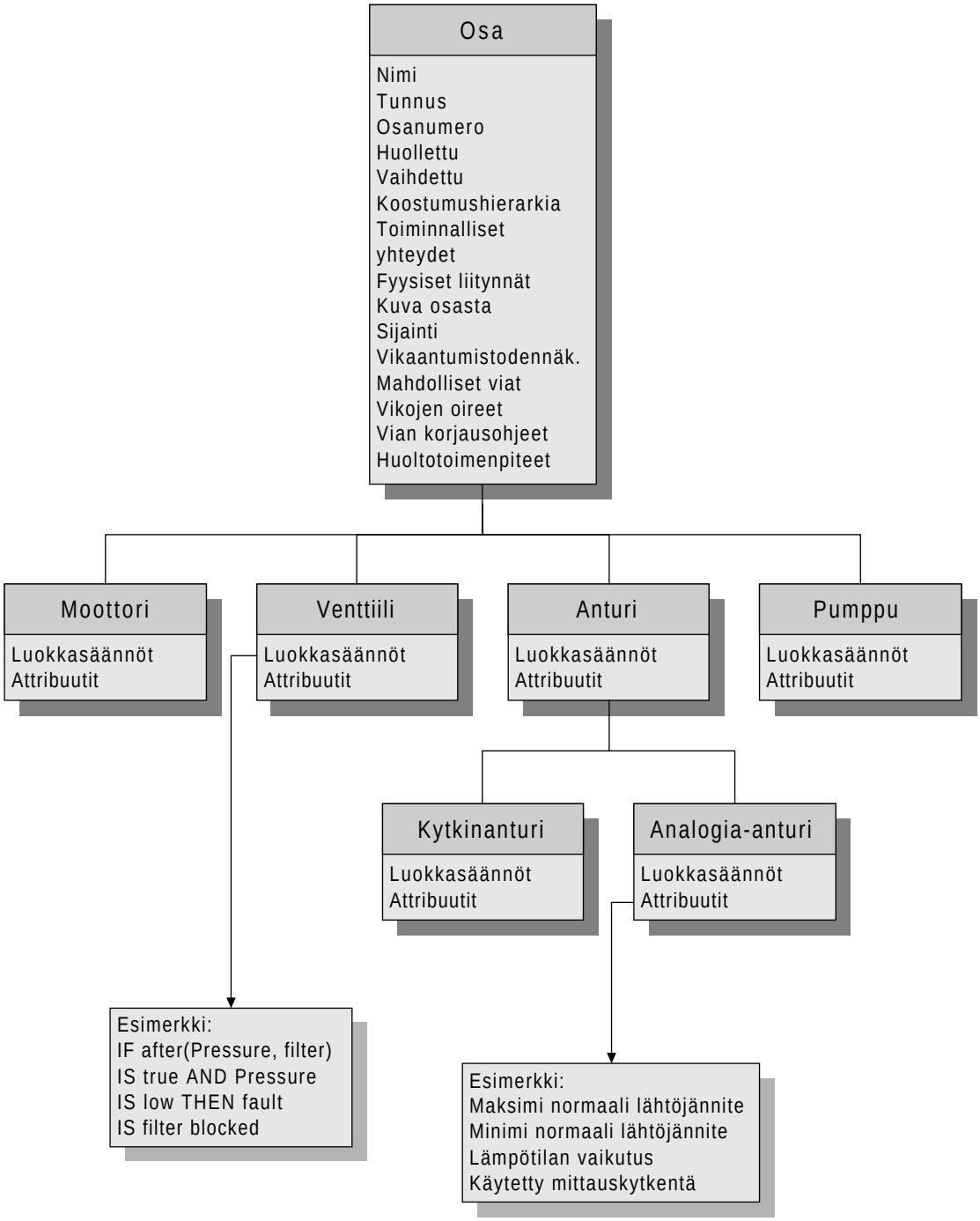
Ominaisuus	Yhteinen tuotetietokanta (käytetty menetelmä)	STEP-standardin määrittelemä tuotemalli	Tietämyspohjaiset CAD-järjestelmät	OOCAD-projektin tuotemalli
Muutokset perinteiseen CAD-suunnittelu-prosessiin	●	●●●●	●●●●●	●●
Käyttöönoton resurssi-vaatimukset	●	●●●●	●●●●●	●●
Suunnittelu-sääntöjen käyttäminen	●	●●●●	●●●●	●
Kattavuus	●●	●●●●	●●●	●
Laite- ja ohjelmisto-riippuvuus	●	●	●●●●●	●●●●
Tiedon hallinnan vaikeutuminen tiedon määrän kasvaessa	●●●●●	●●	●●	●●●●●
Soveltuvuus suunnittelu-tietämyksen kuvaamiseen	●	●●●	●●●●	●
Soveltuvuus mekatronisen laitteen tuotetiedon kuvaamiseen	●●	●●●●	●●●●	●

Kuva 26. Yhteenveto tuotemallinnusmenetelmien ominaispiirteistä ja käyttöönoton vaikutuksista (● = vähäinen, ●●●●● = huomattava).

6.3 TUOTEMALLINNUKSEN HYÖDYNTÄMINEN KUNNONVALVONNASSA

Tuotemallinnuksen käytöllä voidaan saavuttaa huomattavia lisähyötyjä mekatronisen laitteen kunnonvalvonnan ja vikadiagnostiikan näkökulmasta. Oliopohjaisilla tuotemallinnusmenetelmillä on mahdollista järjestelmällinen laitteen kuvaaminen laajemmassa mittakaavassa. Laitteen rakennehierarkia, osien ja osa-alueiden rakenteelliset ja toiminnalliset liitynnät sekä mahdollisiin vikoihin liittyvä tietämys pyritään esittämään mahdollisimman formaalisti. Mikäli kunnonvalvonnassa halutaan käyttää graafista esitystä, jonkintasoinen geometrian kuvaaminen [33] on välttämätöntä.

Kuvassa 27 on esimerkki siitä, millainen mekatronisen laitteen oliopohjaisen tuotemallin luokkahierarkia voi olla.



Kuva 27. Esimerkki mekatronisen laitteen osatason luokkahierarkiasta.

Luokassa “Osa” määritellään osien ominaisuudet attribuutteina, jotka periytyvät kaikille aliluokille. Kullakin aliluokalla voi perittyjen lisäksi olla omia yksilöllisiä attribuutteja. Kaikki ominaisuudet eivät välttämättä periydy. Luokilla voi olla myös sääntöjä, joita voidaan käyttää mahdollisten vikatilanteiden automaattiseen tunnistamiseen [2]. Olioiden ilmentymät eli varsinaiset osien ja komponenttien kuvaukset voidaan tallentaa ja kirjastoida, mikä mahdollistaa tuotemallinnustiedon ja tietämyksen uudelleenkäytön.

Tuotemallinnuksen avulla välitettyä tietämystä ja tietoa on mahdollista hyödyntää esim. seuraavin tavoin kunnonvalvonnassa ja vikadiagnostiikassa:

- Laitteen toiminnan visuaalinen valvonta on mahdollista, jos käytössä on graafinen näyttö.
- Vikapaikannuksessa voidaan käyttää hyväksi osien toiminnallisia ja rakenteellisia riippuvuuksia. Osat on tällä tavoin kytketty toisiinsa, jolloin vikojen syy-yhteyksien selvittäminen helpottuu.
- Käyttäjän on mahdollista nähdä laite kokonaisuutena tuotemallin rakennehierarkian avulla ja siten hahmottaa laitteen toiminta paremmin.
- Huoltohenkilö tai käyttäjä näkee vaivattomasti osien fyysiset sijainnit laitteessa.
- Kaikki tieto osista on saatavilla yhdessä paikassa.
- Tuotemallinnus mahdollistaa online-dokumenttien kuvien ja tekstin yhdistämisen loogiseksi kokonaisuudeksi helpottaen tiedon etsintää.

7 YHTEENVETO

Työssä tarkasteltiin mekatronisen laitteen CAD-suunnittelutiedon ja -tietämyksen esittämistapoja kunnonvalvonnan ja vikadiagnostiikan näkökulmasta. Lisäksi tutkittiin mahdollisuuksia suunnittelutiedon siirtämiseksi ja muokkaamiseksi laitteen sulautetun ohjelmiston osaksi.

Tavoitteena oli CAD-sähkösuunnittelun ja ohjelmistosuunnitteluprosessin välisen rajapinnan tiedonsiirron tehostaminen automatisoimalla suunnittelutiedon siirto ja muokkaus CAD-sähkökuvista C-kielisiksi tietorakenteiksi. Tietorakenteita hyödynnettiin maanalaiseen kaivaukseen tarkoitetun kallioporauslaitteen reaaliaikaisessa kunnonvalvontaohjelmassa, joka piti suunnitella ja toteuttaa. Kunnonvalvontaohjelmalta vaadittiin selkeä ja yksinkertainen tiedon välittäminen hajautettujen liityntämoduulien ja toimilaitteiden tiloista, antureiden arvoista ja sähköjärjestelmän rakenteesta.

Suunnitteluprosessien välisen rajapinnan työvaiheiden automatisointia varten otettiin käyttöön suunnittelutiedolle yhteinen tietokanta. CAD-mallien ja tietokannan välinen tiedonsiirto voidaan toteuttaa CAD-järjestelmien tarjoamien sovelluskehitys-ympäristöjen avulla. Tietokannan ja ohjelmistosovelluksen välistä tiedonsiirtoa varten kehitettiin Windows-ohjelma, joka tuottaa tietokannan tietojen perusteella automaattisesti C-kieliset tietorakenteet sovellusta varten. Tietorakenteita hyödyntävä reaaliaikainen kunnonvalvontasovellus suunniteltiin ja toteutettiin simulaatiomallien avulla PC Windows -ympäristössä. Sulautettava kunnonvalvontaohjelma testattiin kallioporauslaitteen ohjausjärjestelmää vastaavassa testilaitteessa.

Työn tuloksena demonstroitiin menetelmä CAD-sähkösuunnittelutiedon siirtämiseksi automaattisesti ohjelmistosovellusten käyttöön. Lisäksi suunniteltiin ja toteutettiin tavoitteiden mukainen kallioporauslaitteen kunnonvalvontaohjelma, jota voidaan käyttää vikahavainnoinnissa. Työssä esitetyllä menetelmällä vähennetään automatisoinnin myötä ihmisen tekemien virheiden määrää. Sulautettavien ohjelmien ylläpidettävyys paranee, kun CAD-malleihin tehtyjä muutoksia ei enää tarvitse erikseen ohjelmoida sovelluksiin. Toimitusaikojen lyhentäminen ja asiakaskohtaistaminen ovat myös paremmin mahdollisia suunnittelun osa-alueiden välisen tiedonsiirron kehittyessä. Muita etuja ovat vähäiset resurssivaatimukset ja vaikutukset yrityksen CAD-suunnittelun kulkuun. Menetelmä on järjestelmäriippumaton, helppo ottaa käyttöön, eikä aiheuta CAD-suunnitteluprosessin monimutkaistumista.

Esitetty tiedon kuvaus- ja siirtomenetelmä ei kuitenkaan ole riittävän kattava ja ilmaisuvoimainen laajempaan suunnittelutietämyksen hyödyntämiseen kunnon-valvonnassa. Sen avulla ei varsinaisesti tuoteta mitään korkean tason mallia järjestelmästä, vaan ryhmitellään ja muotoillaan tieto ohjelmistosuunnittelijan käyttöön. Tieto saatetaan CAD-järjestelmien esitysmuodosta formaaleiksi tietorakenteiksi, joiden ymmärtäminen ja käyttö on ohjelmistosuunnittelijan

vastuulla. Tiedon esityksen täytyy tällöin olla erittäin hyvin ryhmitettyä ja kuvaavasti nimettyä.

Lisäksi tulee ottaa huomioon tietorakenteiden automaattiset tuottajaohjelmat. Yksinkertaisesta rakenteestaan huolimatta niiden suunnittelu ja ohjelmointi vaatii jonkin verran panostusta. Tätä seikkaa tärkeämmäksi noussee kuitenkin tuottajaohjelmien ylläpito CAD-suunnittelun tiedon esitysformaatin muuttuessa. Mikäli CAD-suunnittelun esitystapoja ei ole kunnolla yhdenmukaistettu, tuottajaohjelmien ylläpito voi muodostaa ongelman. Lisäksi ongelmia seuraa varmasti, jos kaikki suunnittelijat eivät käytä samoja esitystapoja samoille asioille.

Yhdenmukaisuusongelmaan saadaan ratkaisu käyttämällä CAD-suunnittelussa standardoitua tuotemallinnusta. Esimerkiksi STEP-malli on tähän tarkoitukseen sopiva. Tällöinkin jonkinasteinen tuottajaohjelmien ylläpito on tarpeen, mikäli laitteen rakenteessa tapahtuu merkittäviä muutoksia. Tuotemallinnuksen käyttö takaa joka tapauksessa yhdenmukaiset suunnittelusäännöt ja -esitystavat sekä tuottaa havainnollisen mallin laitteesta.

LÄHTEET

1. Bradley, D., Dawson, D., Burd, N. & Loader, A. 1991. Mechatronics: Electronics in products and processes. Chapman and Hall. 510 s.

2. Kurki, M. 1995. Model-based fault diagnosis for mechatronic systems. Espoo: Technical Research Centre of Finland. 116 s. VTT Publications 223.

3. Salminen, V., Verho, A. & Laurila, T. (toim.) 1990. Mekatroniikka 1987 - 1990. Helsinki: TEKES, teknologiaohjelmien kirjasarja. 159 s.

4. Taramaa, J., Seppänen, V., Lintulampi, R. & Miettunen, A. 1994. Automaattisten koneiden ohjausohjelmistojen kokoonpano. Metalliteollisuuden keskusliitto, tekninen tiedotus. 40 s.

5. Ward, P. & Mellor, S. 1985. Structured Development for Real-Time Systems. Vol. 1 - 3. New York, USA: Yourdon Press. 501 s.

6. Lipsett, R., Schaefer, C. & Ussery, C. 1989. VHDL: Elektroniikan kuvaus- ja suunnittelukieli. Boston, USA: Kluwer Academic Publishers. 348 s.

7. Tanskanen, K. 1990. Mekatroniikan järjestelmällinen suunnittelu. Metalliteollisuuden keskusliitto, tekninen tiedotus. 72 s.

8. Viitanen, P., Yli-Paunu, P. & Yli-Pietilä, T. 1992. Mekatronisen laitteen hierarkkinen suunnittelu ja toiminnallinen simulointi. Metalliteollisuuden keskusliitto, tekninen tiedotus. 77 s.

9. Mäntylä, M. 1993. Towards Open Architecture Concurrent Engineering Frameworks. Helsinki University of Technology. 20 s.

10. Ringland, G. & Duce, D. 1988. Approaches to Knowledge Representation: An Introduction. England: Research Studies Press Ltd. 260 s.

11. Adeli, H. 1990. Knowledge engineering: Vol 1, Fundamentals. McGraw-Hill Publishing Company. 354 s.

12. Leiviskä, K., Kaarela, K. & Oksanen, J. 1992. Tietämyksen integrointi ja sulauttaminen prosessinhallinnassa. Teknillinen loppuraportti. Oulu: VTT Elektroniikka & Oulun yliopisto. 136 s.

13. Björk, B. 1990. Rakennuksen tuotemallistandardin kehittämisnäkökohtia. Raportti. Espoo: VTT Yhdyskunta- ja rakennussuunnittelun laboratorio. 15 s.

14. Spaccapietra, S. 1987. Entity-Relationship Approach: Ten Years of Experience in Information Modeling. Proceedings of the Fifth International Conference on Entity-Relationship Approach. Amsterdam, The Netherlands: Elsevier Science Publishers. 557 s.

15. Booch, G. 1991. Object Oriented Design with Applications. The Benjamin/Cummings Publishing Company. 580 s.

16. Mäkilehto, T. 1994. Tamrock Oy:n automatisointijärjestelmän I/O-kuvauksen rakenne ja sen käyttö. Insinöörityö. Riihimäen teknillinen oppilaitos, sähköosasto. 30 s.

17. Tamrock 1994. Electrohydraulic drilling control, SOLO, Operation, adjusting and maintenance. Tamrock Oy. 25 s.

18. Alanen, J. & Virtanen, A. 1994. Ylemmän kerroksen CAN-kommunikointiarkkitehtuurit. VTT tiedotteita 1561. Espoo: Valtion teknillinen tutkimuskeskus. 61 s.

19. Dibble, P. 1988. OS-9 Insights: An Advanced Programmer's Guide to OS-9/68000. Des Moines, USA: Microware. 348 s.

20. Autodesk 1994. AutoCAD Release 13, käsikirja. Autodesk Inc.. 670 s.
Autodesk 1994. AutoCAD Release 13, käskyopas. Autodesk Inc.. 820 s.
Autodesk 1994. AutoCAD Release 12, AutoLISP Programmers Reference. Autodesk Inc.. 252 s.
Autodesk 1994. AutoCAD Release 13, Customization Guide. Autodesk Inc.. 670 s.

21. Microsoft 1994. Microsoft Access Relational Database Management System for Windows, User´s Guide. Microsoft Corporation. 819 s.

22. Microsoft 1994. Microsoft Access Relational Database Management System for Windows, Language Reference. Microsoft Corporation. 819 s.

23. Insoft 1988. Prosa User's Manual. Oulu: Insoft Ky. 237 s.

24. Microsoft 1994. Visual C++ Development System for Windows and Windows NT, Introducing Visual C++. Microsoft Corporation. 436 s.

25. Milenkovic, M. 1987. Operating Systems: Concepts and Design. McGraw-Hill Publishing Company. 568 s.

26. McGilton, H. & Morgan, R. 1983. Introducing the UNIX System. McGraw-Hill Publishing Company. 556 s.

27. Leskelä, J. 1994. Sulautettujen reaaliaikaohjelmistojen visualisointi simulointipohjaisessa testauksessa. Diplomityö. Oulun yliopisto, sähkötekniikan osasto. 62 s.

28. Coad, P. & Yourdon, E. 1990. Object-Oriented Analysis. New Jersey, USA: Prentice-Hall. 232 s.

29. ISO TC184/SC4 1994. ISO 10303 (Standard for the Exhange of Product Model Data). International Organization for Standardisation. (http://elib.cme.nist.gov/nipde/Intro.html)

30. Björk, B. 1990. STEP - tuotetietojen tiedonsiirtostandardi. Raportti. VTT Yhdyskunta- ja rakennussuunnittelun laboratorio. 10 s.

31. Säkkinen, J. 1991. Tietämystekniikka ja CAD. Valokynä 3/91, s. 14 - 16.

32. Pellosniemi, J. 1994. Oliopohjaisten tuotemallinnuspiirteiden lisääminen CAD-järjestelmään. Valokynä 2/94, s. 17 - 21.

33. Helpenstein, H. (Ed.) 1993. CAD Geometry Data Exchange Using STEP. ESPRIT Subseries PDT, Project 2195 CADEX. Research reports, Vol 1. Springer-Verlag. 432 s.

ESIMERKKI AUTOMAATTISESTI CAD-TIEDOSTA TUOTETUSTA OTSIKKOTIEDOSTOSTA

```
#define MODULE_NUMBER                   10
#define COUNTRY_NUMBER                  3
#define IO_NUMBER                       9
#define CONNECTOR_MAX_NUMBER            16



struct MODULE
{
        char *type                                      [ MODULE_NUMBER ];
        char *location                                  [ MODULE_NUMBER ];
        char *number                                    [ MODULE_NUMBER ];
}

        module =
{
        "DIO8"      , "PWM4"  , "DIO8"    , "DIO8"    , "UNI"      ,
        "DIO8"      , "DIO8"  , "DIO8"    , "DIO8"    , "UNI"      ,
        "R1"        , "R1"    , "R2"      , "R2"      , "R3"       ,
        "R3"        , "R4"    , "R4"      , "R5"      , "R5"       ,
        "00"        , "01"    , "02"      , "03"      , "04"       ,
        "05"        , "06"    , "07"      , "08"      , "09"
};



struct INPUT_OUTPUT
{
        char *name                                      [ IO_NUMBER ]
                                                        [ COUNTRY_NUMBER ];
        char *module_number                             [ IO_NUMBER ];
        char *type                                      [ IO_NUMBER ];
        char *symbol                                    [ IO_NUMBER ];
        char *channel_number                            [ IO_NUMBER ];
        char *cable_number                              [ IO_NUMBER ];
        char *connector_numbers                         [ IO_NUMBER ]
                                                [ CONNECTOR_ MAX_NUMBER
];
        char *state                                     [ IO_NUMBER ];
}

        IO =
{
        "Vesihuuhtelu"                                  ,
```

"Water flushing" ,
"Vattenspolning" ,
"Ilmahuuhtelu" ,
"Air flushing" ,
"Luftspolning" ,
"Maatukien liiketieto" ,
"Jacks moving by diesel" ,
"Landfästen rörelse med diesel" ,
"Käynnissä merkkivalo" ,
"Aggregat in running" ,
"Aggregat på gång" ,
"Hätäpysäytetty" ,
"Emergency stopped" ,
"Nödstopp" ,
"Puomin kallistus eteen" ,
"Boom tilting forwards" ,
"Lutning av bomens framåt" ,
"Iskun tuntimittari" ,
"Percussion hourmeter" ,
"Slag timräknare" ,
"Iskun paineanturi" ,
"Percussion pressure transducer" ,
"Tryckgivare för slaget" ,
"Syötön paineanturi" ,
"Feed pressure transducer" ,
"Tryckgivare för matning" ,

"02" ,
"02" ,
"02" ,
"00" ,
"00" ,
"03" ,
"00" ,
"01" ,
"01" ,

"DI" ,
"DI" ,
"DI" ,
"DO" ,
"DO" ,
"DO" ,
"DO" ,
"AI" ,
"AI" ,

"Y9" ,
"Y10" ,
"Y168" ,
"SH82" ,
"H614" ,
"Y22" ,
"P100" ,
"B1" ,
"B3" ,

"0" ,
"1" ,
"2" ,
"0" ,
"4" ,
"0" ,
"2" ,
"0" ,
"1" ,

"1" ,
"2" ,
"3" ,
"4" ,
"5" ,
"6" ,
"7" ,
"8" ,
"9" ,

"1", "2", "3", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ",
"4", "5", "6", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ",
"7", "8", "9", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ",
"10", "11", "12", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ",
"13", "14", "15", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ",
"16", "17", "18", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ",
"19", "20", "21", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ",
"22", "23", "24", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ",
"25", "26", "27", "28", "29", "30", "31", "32", "33", "34", "35", "36",
"37", "38", "39", "40",

"0" ,
"0" ,
"0" ,
"0" ,
"0" ,
"0" ,