Sähkötekniikan korkeakoulu

Automaatio- ja systeemitekniikka

FINNISH

KANDIDAATINTYÖN TIIVISTELMÄ

Tekijä: Ian Tuomi

Työn nimi:

Päivämäärä: Kieli: Suomi Sivumäärä:5+6

Tutkinto-ohjelma:

Vastuuopettaja:

Tämän tutkimuksen kohteena on järjestelmän ohjauslogiikan formaalin kuvauksen muuntaminen standardeita noudattavaksi logiikkakaavioksi.

Avainsanat:

Esipuhe

Otaniemi, 16.3.2010

Ian Tuomi

Sisältö

Tiivistelmä **i**

Esipuhe **iii**

Lyhenteet **v**

1 Johdanto **1**

2 Tehtaan mallit ja mallinnustavat **2**

2.1 Automaatiojärjestelmän logiikka 2

2.2 Logiikkadiagrammit 3

3 Hierarkinen graafinpiirto **4**

3.1 Syklien poisto 4

3.2 Kerrostus 4

3.3 Risteyksien vähentäminen 5

3.4 Poikittainen asettelu 5

Viitteet **6**

Lyhenteet

DCS Distributed Control System
Hajautettu ohjausjärjestelmä

1 Johdanto

Teollisuuden automaatiojärjestelmän suunnittelutyön tulos koostuu kaavioista - laajan järjestelmän tapauksessa jopa tuhansista sellaisista. Kaaviot muodostavat kuvauksen joka mahdollistaa tehtaan toteutuksen, käyttöönoton ja ylläpidon. Kaavioiden piirtäminen käsin on suurissa projekteissa työlästä ja tehotonta. Nykyaikaisessa tehdassuunnittelussa ongelma korostuu. Ratkaisuista on tullut monimutkaisempia ja laatu- ja joustavuusvaatimukset ovat kasvaneet. Lisäksi suunnitteluprojektien toteutusaikavaatimukset ovat jatkuvasti lyhentyneet. [1]

Tämän vuoksi suunnitteluprosessia on pyritty tehostamaan automatisoiduilla suunnittelun apuvälineillä. Suunnittelumetodiikat eivät ole ilmeisistä standardoimiseduista huolimatta yhtenäistyneet, ja yhteistyö alan tutkimuksessa on vähäistä. Vaikka lähestymistavat ovat vaihtelevia, on tehdassuunnittelussa kuitenkin nähtävissä yhtenäinen siirtymä kohti tietomalleja. Automaatiosuunnittelun kannalta tämä tarkoittaa sitä, että suunnittelujärjestelmään syntyy projektin edetessä tehtaan tietomalli johon tehtaan kaikki sähköistys ja instrumentointi automaatiojärjestelmineen ja kenttälaitteineen on merkitty. Erilaiset kaaviot voidaan tuottaa suoraan tietomallin sisältävästä tietokannasta.

Tietomalliin perustuva lähestymistapa vaatii tietomallin laajuuden ja rajoitukset määrittelevän metamallin. Tämä malli ohjaa ja määrittää suunnittelun kohteen lisäksi myös yleisesti suunnittelutöiden kulkua.

Tämän tutkimuksen kohteena on hajautetun ohjausjärjestelmän logiikan tietomallin muuntaminen standardeja noudattavaksi logiikkakaavioksi. Logiikkakaaviot ovat tapa pelkistää syöttöjen ja lähtöjen välillä vallitseva logiikka. Logiikkakaavio on määritelty yksiselitteisesti ja on tulkittavissa suunnatuksi graafiksi, jolloin algoritmillisen graafinpiirron tuloksia voidaan hyödyntää.

Algoritmillinen graafipiirto on perusteellisesti tutkittu ala, jonka tuloksia voidaan käyttää kaikenlaisten suunnattujen kaavioiden piirtoon. Esittelemme Sugiyaman et. al. [2] esittämään hierarkiseen lähestymistapaan perustuvan algoritmin jonka avulla lopullinen kaavio piirretään.

2 Tehtaan mallit ja mallinnustavat

Tehdasmalli on kuvaus joka mahdollistaa tehtaan toteutuksen, käyttöönoton ja ylläpidon. Se on tietomalli joka kuvaa laajuudestaan riippuen tehtaan toimintaa, sen prosessia, organisaatiota, ihmisiä ja näitten aktiviteetteja. Tehdasmalli koostuu pohjimmiltaan tehdasobjekteista, niitten ominaisuuksista ja niitten välisistä relaatioista. [1]

Suunnittelijoilla on työtehtävistään riippuen erilaisia tarpeita tehdasmallin suhteen. Työn kohde on monitahoinen eikä mikään yksi malli pysty kattamaan kaikkia suunniteltavan järjestelmän näkymiä. Tehdasta tulee kuvata useilta eri näkökannoilta erilaisilla tarkkuuksilla kunkin näkymän vaatimuksien mukaisesti. Tehdasmalliin kuuluu siis useita malleja, jotka yhdessä muodostavat kokonaisen systeemin.

Jokaisella mallilla on myös metamalli joka määrittelee millaisia objekteja malli voi sisältää, mitä ominaisuuksia niillä voi tai täytyy olla ja millaisia yhteyksiä objektien välillä on. Metamallin loogiset riippuvuudet määrittelevät suunnittelun työtapoja ja järjestystä. Ne kuvaavat millaiset tietomallit ovat suunnittelujärjestelmän kannalta hyväksyttäviä tehdasmalleja. Ne määräävät myös suunnittelujärjestelmän laajuuden. Metamallin määritteleminen tekee suunnittelutyön tuloksesta ennakoitavan ja mahdollistaa suunnittelun apuvälineiden kehittämisen. Suunnittelutyön formalisoimisella myös suljetaan pois eriäviä tapoja kuvata tehdasta ja mahdollistetaan formalismiin perustuvat työkalut ja suunnittelukäytännöt.

Suunnittelutyössä käytettävät metamallit valitaan siten, etteivät ne ole ristiriidassa ja muodostavat yhdessä toisiaan täydentävän kokonaisuuden. Tällöin metamallien konfiguraatio muodostaa itsessään mallin.

2.1 Automaatiojärjestelmän logiikka

Teollisuuden hajautettujen ohjausjärjestelmien suunnittelussa on havaittu tarpeelliseksi ainakin kolme näkymää [3]: funktionaalinen näkymä, fyysinen näkymä sekä ohjelmistonäkymä.

Näistä ensimmäinen määrittelee järjestelmän toiminnallisuuden. Tätä näkymää määritettäessä voivat eri alojen suunnittelijat osallistua työhön ilman tuntemusta instrumentoinnista tai ohjelmistokehityksestä. Kun päätökset on tehty abstraktilla tasolla, voidaan hankinnat räätälöidä sen mukaisesti ja tulos toteuttaa tehokkaasti. Funktionaalisuuden suunnitteluprosessin tuloksena on usein PI-kaavio joka usein toimii esisuunnittelutietona muille näkymille.

Fyysiseen näkymään suunnitellaan järjestelmän johdotukset, kaapelit, kenttä-

väylät, ohjauskeskukset, toimilaitteiden sijainnit sekä ylipäänsä kaikki mitä vaaditaan järjestelmän fyysiseen käyttöönottoon.

Ohjelmistonäkymä kattaa funktionaalisen suunnittelun toteutuksen ohjelmamuodossa. Ohjelmistotototeutus jää suunnitteluprosessissa harvinaisen vähäiselle formaaliuden asteelle, vaikka pyrkimyksiä siihen suuntaan on ollut.

Projektin edetessä konseptista toteutukseksi huomataan, että tuotetut dokumentit ovat kahdenlaisia. Ne voitaisiin luonnehtia suunnittelutyön sivutulokseksi sekä lopputulokseksi. Suunnittelutyön tavoitteen toteuttava dokumentti saattaa olla ainoa joka näkee päivänvalon. Tässä työssä tarkastellaan tapoja joilla suunnittelutyön tuottamia malleja voidaan muuntaa toisiksi, käyttökelpoisiksi malleiksi kuten logiikkadiagrammeiksi.

2.2 Logiikkadiagrammit

Hajautetun ohjausjärjestelmän logiikan suunnittelulle ja toteutukselle ei ole laajassa käytössä olevaa standardia, pikemminkin joukko standardeja joita käytetään vaihtelevasti

IEC 61131-3 ja myöhemmin sitä täydentänyt IEC 61499 ovat pyrkimyksiä tuottaa avoin standardi joka määrittelee automaatiojärjestelmän ohjelmoinimenetelmät ja on riittävän joustava mihin tahansa tarkoitukseen. Se sisältää 6 eri ohjelmointikieltä, joista tässä työssä käsitellään erityisesti FBDtä.

Logiikkadiagrammit pelkistävät syöttöjen ja lähtöjen välisen logiikan yksinkertaiseksi ja intuitiiviseksi esitykseksi. Vaikka ohjausjärjestelmän ohjelmointi toteutettaisiinkin muunlaisella esityksellä, esimerkiksi tikapuukaaviolla, on silti usein helpompaa ymmärtää ohjelman rakenne logiikkadiagrammien avulla. Lisäksi logiikkadiagrammit helpottavat tehtaan toiminnan ymmärrystä suunnittelu- ja rakennusvaiheen lisäksi tehtaan ylläpidon ja huollon yhteydessä ja ovat siksi tärkeä osa tehtaan lopullista dokumentaatiota.

Funktioblokit kuvaavat toisistaan erillisiä itsenäisiä laskennallisia yksiköitä ja niitten relaatioita toisiinsa. Funktioblokki voi tilastaan riippuen lähettää tapahtumia jotka vaikuttavat muitten blokkien toimintaan. Sen tekemien laskutoimitusten tulos riippuu sen vasemmalta puolelta tulevista syöttötiedoista ja sen suorittaman laskennan tulokset lähetetään blokin oikealta puolelta. Toisiinsa liitetyt funktioblokit muodostavat funktioblokkiverkon, joka määrittelee laajemman toiminnallisuuden.

3 Hierarkinen graafinpiirto

Hierarkisen lähestymistavan kerrostettujen suunnattujen graafien piirtämiseen esittivät Sugiyama et. al vuonna 1981 [2]. Tämä lähestymistapa on osoittautunut tehokkaaksi ja on ollut jatkokehityksen- ja tutkimuksen kohteena [4].

Hierarkinen lähestymistapa koostuu kolmesta vaiheesta: kerrostamisesta, risteämisten vähentämisestä ja poikittaisasettelusta. Lisäksi piirrettävän graafin täytyy olla etukäteen valiidisti muotoiltu ja syklitön.

Automaatiokaavioiden suoritusjärjestys on määritelty standardissa IEC 61131-3 ylhäältä alas ja vasemmalta oikealle. Lisäksi funktioblokkia ei ajeta ennen kuin kaikki sitä edeltävät funktioblokit on suoritettu. Suoritusjärjestysvaatimus asettaa myös toteutukselle rajoituksia. Jos tätä ei oteta huomioon, saattaa graafin algoritmillinen piirto johtaa vääränlaiseen kaavioon.

Tämä työ olettaa että eri ohjelmien suoritusjärjestys niillä kohdin kuin se on olennaista on tiedossa ja merkittynä tietokantaan. Myös mahdollisten takaisinkytkentöjen tapauksessa täytyy olla huolellinen, jottei syklien poistovaiheessa kaavion järjestys muutu dramaattisesti.

Työvaiheista kerrostus ja risteyksien vähennysvaihe saattavat muuttaa suoritusjärjestystä. Tällöin algoritmin toimintaa pitää yksinkertaisesti rajoittaa järjestysherkkien blokkien osalta. Koska algoritmit ovat heuristisia, ovat siihen tehtävät yksinkertaiset rajoitukset helppoja toteuttaa.

3.1 Syklien poisto

Jos graafissa on syklejä, eli graafi ei ole suunnattu, käännetään syklin aiheuttavien lankojen suunta ja tehdyt muutokset merkitään muistiin. Graafinpiirron lopuksi syklit käännetään takaisin oikein päin.

3.2 Kerrostus

Kerrostetuksen graafin tulee olla olla kompakti. Tämä tarkoittaa että sen leveys ja pituus ovat pieniä ja kerrosten välinen etäisyys on vakio. Graafin pituuden alaraja on pisimmän yhtenäisen ketjun pituus. Erilaisilla kerrostusmenetelmillä graafin pituus voidaan minimoida leveyden kustannuksella tai siitä voidaan tehdä mahdollisimman kapea pituuden kustannuksella.

Sekä leveyden että pituuden minimointi samanaikaisesti on rinnastettavissa multiprosessoriajastusongelmaan ja siten NP- täydellinen ongelma. [5] Multiprosesso-

riongelmaa varten suunniteltu algoritmi, Coffman-Graham-kerrostusmenetelmä tarjoaa tällöin ratkaisun tähän ongelmaan.

3.3 Risteyksien vähentäminen

Risteyksien vähentäminen ei perustu solmujen tarkkoihin sijainteihin, vaan niitten keskinäiseen järjestykseen. Ongelma on siis luonteeltaan kombinatorinen eikä geometrinen, mikä huomattavasti yksinkertaistaa ongelman ratkaisemista. Tästä helpotuksesta huolimatta kyseessä on NP-täydellinen ongelma siinäkin tapauksessa että kerroksia no vain kaksi. [6]

Lähestymistapoja risteyksien vähentämiseen on vierekkäisten noodien paikkojen vaihtelu keskenään sekä mediaanimenetelmä, jossa noodi asetetaan paikalle, joka on siihen yhtyneiden solmujen puolivälissä.

Monet ovat tutkineet erilaisia risteysalgoritmeja on tutkittu laajalti. Yhtä selkeästi parasta menetelmää ei ole kuitenkaan löytynyt Sen sijaan paras lähestymistapa on osoittautunut algoritmeja yhdisteleväksi heuristiseksi hybridialgoritmiksi.

Tässä menetelmässä solmut uudelleenjärjestetään joka iteraation yhteydessä mediaanimenetelmällä. Sen jälkeen luodaan joukko lähes identtisiä järjestelyitä joissa esiintyy paikallisia transpositioita Lopulta valitaan paras tulos seuraavaa iteraatiota varten. Ratkaisua voisi luonnehtia tietynlaiseksi geneettiseksi algoritmiksi.

3.4 Poikittainen asettelu

Solmujen poikittainen asettelun avulla solmujen välisten lankojen käännösten määrää pyritään vähentämään jotta kaavion luettavuus paranee. Risteyksien vähentämisvaiheessa saavutettu kombinatorinen ratkaisu ja sitä myötä risteysmäärä säilyy. Poikittainen asettelu on tärkeä lopputuloksen luettavuuden kannalta. Asettelun merkitys korostuu kun lankoja on paljon. Jokainen mutka langassa aiheuttaa ylimääräisen rasitteen ihmisen hahmotuskyvyssä.

Viitteet

[1] Suomen automaatioseura ry. Automaatiosuunnittelun prosessimalli: Yhteiset käsitteet verkottuneen suunnittelun perustana. Technical report, Suomen automaatioseura ry, 2007.

[2] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *Systems, Man and Cybernetics, IEEE Transactions on*, 11(2):109–125, 1981.

[3] M. Marcos and E. Estevez. Model-driven design of industrial control systems. In *Computer-Aided Control Systems, 2008. CACSD 2008. IEEE International Conference on*, pages 1253–1258. IEEE, 2008.

[4] E. R. Gansner, E. Koutsofios, S. C. North, and K. P. Vo. A technique for drawing directed graphs. *Software Engineering, IEEE Transactions on*, 19(3):214–230, 1993.

[5] G. D. Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Layered Drawings of Digraphs*. Graph drawing: algorithms for the visualization of graphs. Prentice Hall PTR, 1998.

[6] M. R. Garey and D. S. Johnson. Crossing number is np-complete. *SIAM Journal on Algebraic Discrete Methods*, 4(3):312–316, 1983.