Ian Tuomi

Teollisen ohjausjärjestelmän logiikkakuvauksen muuntaminen esteettiseksi kaavioksi

Automaatio- ja systeemitekniikan laitos

Kandidaatintyö

Espoo 6.12.2012

Vastuuopettaja:

DI Marek Matusiak

Työn ohjaaja:

DI Mika Strömman

Aalto-yliopisto
Sähkötekniikan korkeakoulu

Sisältö

Lyhenteet

DCS Distributed Control System
Hajautettu ohjausjärjestelmä

DFS Depth-first search
Syvyyssuuntainen läpikäynti

FBD Function Block Diagram
IEC 61131-3 standardin määrittelemä ohjelmointikieli

IEC International Electrotechnical Commission,
Kansainvälinen sähköalan standarointiorganisaatio

NP Nondeterministic Polynomial time
Epädeterministisellä Turingin koneella polynomiaalisessa
ajassa ratkeavien ongelmien joukko

PI Prosessien instrumentointi

PLC Programmable Logic Circuit
Ohjelmoitava logiikkapiiri

XML Extensible Markup Language
Merkintäkieli johon voidaan sisällyttää ja jolla voidaan jäsentää tietoa

XSL Extensible Stylesheet Language
Kieliperhe, jolla voidaan määritellä XML-formaatteja tai tehdä XML-muunnoksia

1 Johdanto

Teollista automaatiojärjestelmää suunnitteltaessa tuotetaan suuri määrä erilaisia dokumentteja. Ne yhdessä muodostavat kuvauksen, joka mahdollistaa järjestelmän toteutuksen, käyttöönoton ja ylläpidon. Dokumentaation laatiminen käsin on suurissa projekteissa työlästä ja tehotonta. Nykyaikaisessa automaatio- ja instrumentointisuunnittelussa ongelma korostuu [1]. Ratkaisuista on tullut monimutkaisempia, ja laatu- ja joustavuusvaatimukset ovat lisääntyneet. Lisäksi vaaditut toteutusajat ovat jatkuvasti lyhentyneet.

Johtuen suunnitteluprosessiin kohdistuneista vaatimuksista, työtä on pyritty tehostamaan automatisoiduilla apuvälineillä. Metodiikka ei kuitenkaan ole ilmeisistä standardoimiseduista huolimatta yhtenäistynyt. Automaatiosuunnittelun tutkimuskenttä on lisäksi hajanainen, ja yhteistyö on vähäistä. Vaikka lähestymistavat ovat vaihtelevia, suunnittelussa on nähtävissä yhtenäinen siirtymä kohti jaettuja tietomalleja. Automaatiojärjestelmien näkökulmasta jaettujen tietomallien käyttö tarkoittaa, että suunnittelujärjestelmään tallentuu projektin edetessä kaikki sähköistys ja instrumentointi automaatiojärjestelmineen ja kenttälaitteineen. Erilaiset kaaviot voidaan usein tuottaa suoraan tietomallin sisältävästä tietokannasta.

Tietomalliin perustuva lähestymistapa vaatii sen laajuuden ja rajoitukset määrittelevän metamallin. Se ohjaa ja määrittää suunnittelun kohteen lisäksi myös yleisesti suunnittelutöiden kulkua.

Tämän kandidaatintyön kohteena on ohjausjärjestelmän logiikan muuntaminen standardeja noudattavaksi kaavioksi. Logiikkakaavio on tapa pelkistää järjestelmän eri osien syöttöjen ja lähtöjen välillä vallitseva logiikka visuaaliseen ja helposti ymmärrettävään muotoon. Se on määritelty yksiselitteisesti ja on tulkittavissa suunnatuksi graafiksi, jolloin algoritmillisen graafinpiirron tuloksia voidaan hyödyntää.

Algoritmillinen graafinasettelu on perusteellisesti tutkittu ala, jonka tuloksia voidaan käyttää kaikenlaisiin suunnattuihin graafeihin. Tässä työssä eritellään Sugiyaman ym. [2] esittämään hierarkiseen lähestymistapaan perustuva, logiikkakaavioiden kaltaisten tietovirtakaavioiden asetteluun soveltuva algoritmi.

Työssä on myös kokeellinen osio, jossa laadittua algoritmia käytetään konkreettisen kaavion asetteluun. Lopuksi pohditaan tapoja sisällyttää menetelmä käytännön suunnittelutyöhön.

2 Tehtaan mallit

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

Suunnittelijoilla on työtehtävistään erilaisia tarpeita tehdasmallin suhteen. Työn kohde on monitahoinen, eikä mikään yksi esitystapa 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 kokonaismallin.

Jokaisella mallilla on myös metamalli. Se määrittelee millaisia objekteja se voi sisältää, mitä ominaisuuksia sillä 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 ja määräävät sen laajuuden. Metamallin määrittäminen tekee työn tuloksesta ennakoitavan ja mahdollistaa tehokkaiden apuvälineiden kehittämisen. Määrittelyllä myös suljetaan pois eriäviä tapoja kuvata tehdasta ja mahdollistetaan määrittelyyn perustuvat työkalut ja suunnittelukäytännöt. Tällaisia ovat esimerkiksi uudelleenkäytettävää koodia sisältävät kirjastot ja erilaiset mallimuunnokset.

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

2.1 Automaatiojärjestelmän kuvaukset

Teollisuuden hajautettujen ohjausjärjestelmien suunnittelussa on havaittu tarpeellisiksi ainakin kolme kuvausta: toiminnallinen, fyysinen sekä ohjelmistollinen [3].

2.1.1 Toiminnallinen kuvaus

Toiminnallinen kuvaus määrittelee säätöjärjestelmän toiminnan tavalla, joka toteutettuna täyttää sille asetetut käyttäjävaatimukset. Se kuvaa järjestelmän toimintaa,

muttei määrää sen tarkkaa teknistä toteutusta. [4]

Tätä näkymää määrittäessä 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. Tällaisen määrittelyn avulla on mahdollista käyttää formaaleja verifiointimenetelmiä joitten avulla voidaan osoittaa, että järjestelmän suunnitelma toteuttaa sille asetetut vaatimukset. Näin suunnitelman oikeellisuus voidaan osoittaa ennen investointeja toteutukseen. [1]

2.1.2 Fyysinen näkymä

Fyysiseen näkymään suunnitellaan järjestelmän johdotukset, kaapelit, kenttäväylät, ohjauskeskukset, toimilaitteiden sijainnit sekä ylipäänsä kaikki mitä vaaditaan järjestelmän fyysiseen käyttöönottoon. Sen käsittely ei kuulu tämän työn laajuteen.

2.1.3 Ohjelmistonäkymä

Ohjelmistonäkymä kattaa funktionaalisen suunnittelun toteutuksen ohjelmamuodossa. Asiakkaalla on yleensä omat toiveensa sen suhteen, millaiseen muotoon ohjelma tehdään. Ohjelmiston toteuttamiseen ei pitäisi liittyä enää suunnittelutyötä - toisin sanoen toiminnallisen kuvauksen määrittelyjen täytyy olla riittävän tarkkoja jotta niitten mukaan tehdyssä ohjelmassa ei ole sen toimintaan vaikuttavia tulkinnanvaraisuuksia.

Ohjelmistotototeutuksen työtavat jäävät usein määrittelemättömiksi ja muusta suunnittelutyöstä irrallisiksi, vaikka pyrkimyksiä toteutustapojen nitten integroimiseksi suunnitteluprosessiin entistä vahvemmin on ollut. Toteutustapa jää yksittäisten suunnittelijoiden määriteltäväksi, mistä seuraa että yhtenäistä työtapaa harvoin on. Toteutetun ohjelmakoodin uudelleenkäyttö jää myös tyypillisesti vähäiselle asteelle. [5]

2.2 Logiikan kuvaukset

Logiikan kuvaukset helpottavat tehtaan toiminnan ymmärrystä suunnittelu- ja rakennusvaiheen lisäksi tehtaan n ja huollon yhteydessä ja ovat siksi tärkeä osa tehtaan lopullista dokumentaatiota. Teollisessa automaatiosuunnittelussa sallitut logiikan esitystavat on määritelty erityisesti selkeyttä, turvallisuutta ja ennustettavaa toimintaa silmälläpitäen. Joskus loogista järjestelmää voidaan kuvailla kertomalla esimerkinomaisesti sen toiminnasta eri olosuhteissa. Tällainen lähestymistapa ei kuitenkaan yleensä ole riittävä. Lisäksi määritellään yleensä ainakin muuttujia, säätöpiirejä, sekvenssejä sekä niitten välisiä kytkentöjä.[1]

Jatkuva prosessiohjaus voidaan esittää selkeästi järjestelmän PI-kaavioissa, mutta diskreetin ohjauksen esittämiseen tarvitaan erilaisia esityksiä, kuten logiikkadiagrammeja. [6]

Logiikkadiagrammit kuvaavat syöttöjen ja lähtöjen välistä logiikkaa. Määrittelyn mukaisia kaavioita ei kuitenkaan ole välttämättä helppoa laatia.

2.2.1 Toimintakaaviot

Toimintakaavio on funktionaalisen näkymän toteutus, joka tavallisesti valmistellaan järjestelmän suunnittelun varhaisessa vaiheessa prosessikaaviosta. Se määrittelee ohjausjärjestelmän toiminnot ja sitä pidetään ajantasaisena järjestelmän suunnittelun edetessä. Se toimii suunnittelun apuna ja on lopulta osa lopullisen järjestelmän ohjeistusta.

Nykyaikaisessa automaation ohjelmistosuunnittelussa käytetään usein ohjelmointikieliä jotka ovat muodoltaan lähellä toiminnallisia logiikan kuvauksia ja dokumentoivat itsensä hyvin. T'ämän vuoksi toiminnallinen kuvaus usein jätetään tekemättä, jolloin syntyy säästöjä. [1]

2.2.2 Ohjelmakoodi

Ohjelmakoodi on ohjelmanäkymän toteutus. IEC 61131-3 on laajassa käytössä oleva avoin ohjelmakoodin standardi joka määrittelee automaatiologiikan ohjelmointimenetelmät ja pyrkii olemaan riittävän joustava mihin tahansa tarkoitukseen. [7]

Se sisältää viisi erilaista ohjelmointikieltä, joista tässä työssä käsitellään erityisesti FBD:tä.

FBD-kieli koostuu toimilohkoista ja niiden välisistä langoista. Toimilohkot kuvaavat toisistaan erillisiä itsenäisiä laskennallisia yksiköitä. Langat kuvaavat niitten relaatioita toisiinsa. Toimilohko voi tilastaan riippuen lähettää tapahtumia jotka vaikuttavat muitten toimilohkojen toimintaan. Sen tekemien laskutoimitusten tulos riippuu sen vasemmalta puolelta tulevista syöttötiedoista ja sen suorittaman laskennan tulokset lähetetään blokin oikealta puolelta. Toisiinsa liitetyt toimilohkot muodostavat verkon, joka määrittelee laajemman toiminnallisuuden. [8]

Toimilohkojen muokkaamiseen on olemassa useita työkaluja. Näistä mainitsemisen arvoisia ovat ainakin FBDK, 4DIAC ISaGRAF ja NxtControl. Ne eivät kuitenkaan tarjoa kaavioiden asetteluun kuin yksinkertaisia työkaluja, jos ollenkaan.

Automaatiokaavioiden suoritusjärjestys etenee standardin IEC 61131-3 mukaan vasemmalta oikealle. Lisäksi toimilohkoa ei ajeta ennen kuin kaikki sitä edeltävät toimilohkot on suoritettu. Keskinäisen suoritusjärjestyksen täytyy myös tulla esille toteuttavassa ohjelmistossa. Suoritusjärjestys kulkee usein ylhäältä alas, mutta tätä ei voi olettaa aina todeksi. [7]

Suoritusjärjestysvaatimus asettaa seuraavassa kappaleessa tarkasteltavalle graafinpiirrolle rajoituksia. Jos niitä ei oteta huomioon, saattaa graafin algoritmillinen piirto johtaa vääränlaiseen kaavioon. Mahdollisten takaisinkytkentöjen tapauksessa vaaditaan huolellisuutta, jottei syklien poistovaiheessa kaavion järjestys muutu.

2.2.3 Logiikan tietokantakuvaukset

Logiikka tallennetaan jaettuun suunnittelumalliin, johon kaikilla järjestelmän suunnittelijoilla on yhteys. Tapa jolla logiikka tallennetaan täytyy olla yhdenmukainen, jotta suunnittelutyössä voidaan tehdä tehokasta yhteistyötä työpaikalla jaettujen yhtenäisten suunnittelun apuvälineiden avulla. Lisäksi kuvauksen tulee olla formaali jotta logiikkaa voitaisiin käsitellä algoritmillisesti. Formaalilla tarkoitetaan tietynlaista, tarkkaan määritetyn syntaksin mukaista kuvausta. Tällöin tulee myös mahdolliseksi toteuttaa helposti tietoa käsitteleviä muunnosalgoritmeja. Muunnoksella tarkoitetaan jonkin tiedon kuvauksen muuttamista toiseksi kuvaukseksi.

Standardin IEC 61131-3 mukaisten ohjelmointikielten kuvaamiseen on ehdotettu

PLCopen-standardia. Sen mukainen logiikan XML-muotoinen kuvaus voi sisältää toimilohkon toiminnallisuuden kannalta välttämättömien tietojen lisäksi toimilohkon sijainnin, koon ja jopa lankojen reititykset. Esitys voidaan tällöin laatia siten, että se on tuotettavan kaavion asettelun kannalta täysin yksiselitteinen. Tällaista kuvausta voidaan silloin pitää dokumentaation lopullisena muotona. [9]

Kun logiikkakuvauksen muoto on määritelty tarkasti, voidaan määritelmän mukaisesti laadittu logiikkakuvaus muuntaa määritelmään nojaavien sääntöjen perusteella. Monet automaatiologiikan suunnitteluohjelmistot pystyvät muuntamaan käyttämänsä kuvauksen PLCopen-muotoiseksi tiedoksi, jolloin suunnittelutietoa voidaan siirtää eri järjestelmien kesken. Erilaisia XML-kuvauksia on helppoa laatia ja niitten sisältämien tietojen pohjalta voidaan tuottaa minkä tahansa muotoisia dokumentteja käyttäen hyväksi XSL-muunnoksia [10]. Eri XML-pohjaisia tiedostoformaatteja käyttävien suunnitteluvälineiden välinen integraatio on tällöin helposti toteutettavissa.

Olemassaolevien suunnitteluohjelmistojen varaan rakennettu suunnittelujärjestelmä voi kuitenkin kohdata haasteita, jos luotetaan liikaa ohjelmien standardinmukaisuuteen. Valmistajat jättävät usein standardinmukaisiksi kutsutuista ohjelmistoista määrittelyn mukaisia ominaisuuksia toteuttamatta. Tämä siitä, että ohjelma on määritelty standardinmukaiseksi, jos se toteuttaa siitä jonkin osan ja mainitsee mitä se jättää toteuttamatta. [5]

3 Yhteenveto

Teollisen automaatiojärjestelmän suunnittelu- ja toteutusprosessi on monimutkainen, kallis ja pitkä. Siinä käytettäviä työvälineitä tehostamalla työ nopeutuu, ja voidaan saavuttaa merkittäviä säästöjä.

Työvälineiden tehostamista on akateemisissa julkaisuissa kartoitettu runsaasti. Tutkimukset ovat kuitenkin olleet usein turhan abstrakteja ja vieraannuttavia teollisille toimijoille. Tämä työ on pyrkinyt tarjoamaan ratkaisun, jolla suunnittelutyötä voidaan konkreettisesti tehostaa, ja joka voidaan helposti integroida olemassaoleviin järjestelmiin.

Työssä on osoitettu yhteys suunnattujen graafien ja logiikkadiagrammien välillä, sekä todettu niitten asetteluun kehitettyjen algoritmien käyttökelpoisuus teollisuuden automaatiosuunnittelussa. Esitelty algoritmi tuottaa hyviä tuloksia, ja on muunneltavissa edelleen erilaisiin tarpeisiin.

Vaikka idea olisi hyvä ja käytännöllinen, se ei välttämättä ole kannattava. Tästä johtuen on paljon hyviäkin ideoita, joita ei toteuteta. Jotta yritys voi varmistaa kilpailukykynsä myös tulevaisuudessa, tulee sen arvioida jatkuvasti toteutuskelpoisten kehitysprojektien kannattavuutta, ja toteuttaa niitä harkintansa mukaan.

Automaatiojärjestelmien suunnittelu- ja ohjelmointityö on muuttunut nopeasti viime vuosikymmeninä, ja on todennäköistä että se muuttuu edelleen.

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] R. Ajo. *Laatu automaatiossa: parhaat käytännöt*. Suomen automaatioseura, 2001.

[5] J. Kääriäinen. Ohjelmakirjaston hyödyntäminen automaatiojärjestelmässä. 2008.

[6] F. Meier, P. F. D. PFD, and L. Numbering. Control system documentation. *A Guide to the Automation Body of Knowledge*, page 75, 2006.

[7] International Electrotechnical Commission. Programmable controllers-part 3: Programming languages. *IEC Publication*, pages 1131–1133, 1993.

[8] K. H. John and M. Tiegelkamp. *IEC 61131-3: Programming Industrial Automation Systems: Concepts and Programming Languages, Requirements for Programming Systems, Decision-making Aids*. Springer, 2010.

[9] E. van der Wal. Introduction into iec 1131-3 and plcopen. In *The Application of IEC 61131 to Industrial Control: Improve Your Bottom Line Through High Value Industrial Control Systems (Ref. No. 1999/076), IEE Colloquium on*, pages 2/1–2/8. IET, 1999.

[10] J. Clark. Xsl transformations (xslt). *World Wide Web Consortium (W3C).URL http://www.w3.org/TR/xslt*, 1999.

[11] 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.

[12] H. Purchase, R. Cohen, and M. James. Validating graph drawing aesthetics. In *Graph Drawing*, pages 435–446. Springer, 1996.

[13] J. N. Warfield. Crossing theory and hierarchy mapping. *Systems, Man and Cybernetics, IEEE Transactions on*, 7(7):505–523, 1977.

[14] M. J. Carpano. Automatic display of hierarchized graphs for computer-aided decision analysis. *Systems, Man and Cybernetics, IEEE Transactions on*, 10(11):705–715, 1980.

[15] G. Sander. Graph layout through the vcg tool. In *Graph Drawing*, pages 194–205. Springer, 1995.

[16] L. Klauske, C. Schulze, M. Spönemann, and R. von Hanxleden. Improved layout for data flow diagrams with port constraints. *Diagrammatic Representation and Inference*, pages 65–79, 2012.

[17] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972.

[18] P. Eades, X. Lin, and W. F. Smyth. A fast and effective heuristic for the feedback arc set problem. *Information Processing Letters*, 47(6):319–323, 1993.

[19] 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.

[20] G. Sander. A fast heuristic for hierarchical manhattan layout. In *Graph Drawing*, pages 447–458. Springer, 1996.

[21] G. Sander. Layout of directed hypergraphs with orthogonal hyperedges. In *Graph Drawing*, pages 381–386. Springer, 2004.

[22] M. R. Garey and D. S. Johnson. *Computers and intractability*, volume 174. Freeman San Francisco, CA, 1979.

[23] B. Berger and P. W. Shor. Approximation alogorithms for the maximum acyclic subgraph problem. In *Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, pages 236–243. Society for Industrial and Applied Mathematics, 1990.

[24] E. G. Coffman and R. L. Graham. Optimal scheduling for two-processor systems. *Acta Informatica*, 1(3):200–213, 1972.

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

[26] M. Jünger and P. Mutzel. *2-layer straightline crossing minimization: Performance of exact and heuristic algorithms.* MPI Informatik, Bibliothek & Dokumentation, 1996.

[27] L. K. Klauske and C. Dziobek. Improving modeling usability: Automated layout generation for simulink. In *Proceedings of the MathWorks Automotive Conference (MAC'10)*, 2010.