SeAMK
SEINÄJOEN AMMATTIKORKEAKOULU
SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

Markus Soisalo

UML-mallinnuksen hyödyntäminen asiakastarpeen määrittelyssä ja PLC-ohjelmoinnissa

Opinnäytetyö
Kevät 2024
Insinööri (AMK), Automaatiotekniikka

SEINÄJOEN AMMATTIKORKEAKOULU

Opinnäytetyön tiivistelmä

Tutkinto-ohjelma: Insinööri (AMK), Automaatiotekniikka

Suuntautumisvaihtoehto: Koneautomaatio

Tekijä: Markus Soisalo

Työn nimi: UML-mallinnuksen hyödyntäminen asiakastarpeen määrittelyssä ja PLC-ohjelmoinnissa

Ohjaaja: Matti Panula

Vuosi: 2024 Sivumäärä: 38 Liitteiden lukumäärä: 1

Opinnäytetyön tehtiin insinööritoimisto Caplan Oy:lle. Työssä oli kaksi päätavoitetta, joista ensimmäinen oli selvittää, kuinka UML-mallinnusta hyväksikäyttäen voitaisiin parantaa asiakastarpeen määrittelyä automaatioprojektissa. Toimeksiantajalla oli toisinaan ollut haasteita epäselvien asiakastarpeiden kanssa. Toisena tavoitteena oli tutkia, kuinka UML-mallinnusta voitaisiin hyödyntää PLC-projektin dokumentoinnissa sekä ylläpidettävyydessä, jolloin projektikoodin sisältö ja rakenne ymmärrettäisiin nopeasti projektin jälkeenkin.

Toimeksiantajan automaatio-osastolle suoritettiin kyselytutkimus, jonka avulla perehdyttiin siihen, kuinka epäselvä asiakastarpeen määrittely on vaikuttanut työskentelyyn, ja miten asiaa voitaisiin yrittää helpottaa. Ratkaisuna vastauksissa ehdotettiin uutta työkalua, jonka avulla asiakkaan tarpeet saataisiin paremmin esiin.

Ratkaisuna asiakas tarpeen määrittämisen haasteisiin sekä kyselytutkimuksen vastauksien toiveeseen uudesta työkalusta asiakastarpeen määrittelyyn luotiin UML-tilakaaviot asiakkaan toimintakuvauksen perusteella automaatiolaitteesta yleiskielisillä termeillä. Lisäksi luottiin valmiista PLC-projektista UML-luokkakaavio, jonka avulla saatiin ratkaistua tämän opinnäytetyön toinen tavoite.

Opinnäytetyön tuloksena oli uusi työkalu toimeksiantajan käyttöön, työkalulla pystytään helpottamaan toimeksiantajan haasteita. Keskeinen havainto työn aikana oli, kuinka paljon monimutkaisten asioiden visualisointi helpottaa asian ymmärtämistä.

[1] Asiasanat: UML, tehokkuus, ohjelmointi, mallintaminen.

SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

Thesis abstract

Degree programme: Bachelor of Engineering, Automation Engineering

Specialisation: Machine Automation

Author: Markus Soisalo

Title of thesis: Utilizing UML modeling to define customer needs and to ease PLC programming.

Supervisor: Matti Panula

Year: 2024 Number of pages: 38 Number of appendices: 1

The thesis was made for Caplan Oy and it had two main goals. The first one was to study how UML could be used to define customer needs, and the second main goal was to explore how UML-modelling could be used in PLC-projects to document and maintain the PLC-code.

A survey was made at the automation department of Caplan to gain information on how unsuccessful definition of customer needs impacts the work and what kind of solutions would be desired. In the responses of the survey a new tool was suggested.

As a solution to the first goal of the thesis and to meet the needs expressed in the survey responses, a new tool for defining customer needs was created. The new tool, UML, was used to create status diagrams according to customer's operational description of the automation device, aiding in identifying their needs. In addition, a UML class diagram was created from the finished PLC project, which helped to solve the second goal of the thesis.

As the result there was a new tool for the client's use, which can ease the client's challenges. The main finding of the thesis was how the visualizing of complex things makes them easier to understand.

[1] Keywords: UML, efficiency, programming, modelling.

SISÄLTÖ

Kuvio- ja taulukkoluettelo

Käytetyt termit ja lyhenteet

Elementti	UML-mallinuksessa käytetään termejä ”elementti” ja ”piirroselementti”, joilla tarkoitetaan mallin/kaavion yhtä ”rakennuspalikkaa”.
OMG	Object Management Group on kansainvälinen organisaatio, joka kehittää ja ylläpitää avoimia tietotekniikan alan standardeja. OMG koostuu useista organisaatioista.
Pseudotila	UML-mallinnuksessa käytetty termi apuvälineestä, joka viittaa objektin tilaan, joka ei ole varsinainen tila, kuten aloitus- ja lopetustila.

1 JOHDANTO

1.1 Työn tausta

Tämän opinnäytetyön toimeksiantaja on suunnittelutoimisto Caplan Oy. Työn taustalla on ajatus tehostaa automaatioprojektien asiakastarpeen määrittelyä, sekä PLC-koodin dokumentointia ja ylläpidettävyyttä. Nykytilanteessa asiakkaan kanssa on toisinaan vaikeuksia päästä yhteisymmärrykseen, mitä asiakas haluaa, ja ennen kaikkea mitä automaatioprojektiin tarvitaan sen onnistumiseksi. Lisäksi uuden suunnittelijan voi olla vaikea päästä nopeasti kiinni monimutkaisen PLC-projektin rakenteeseen ja toimintaan.

1.2 Työn tavoite

Opinnäytetyön tavoitteena on tutkia ja selvittää, kuinka Unified Model Language (UML) -mallinnusta voidaan soveltaa asiakastarpeen määrittelyssä, ja mikä sovellus olisi optimaalisin edellä mainitun tavoitteen saavuttamiseksi etenkin edellä mainitun tarpeen toiminnallisen puolen osalta. Lisäksi halutaan tutkia, kuinka UML-kaaviota voitaisiin hyödyntää PLC-koodin dokumentoinnissa ja ylläpidettävyydessä.

1.3 Työn rakenne

Työn teoriaosuus aloitetaan perehtymällä UML-mallinnukseen, ja tarkemmin kahteen UML-kaaviotyyppiin. Teoriaosuus jatkuu käyden läpi ohjelmoitavaa logiikkaa sekä PLC-ohjelmointia. Näiden jälkeen siirrytään itse tutkimusosuuden pariin, joka aloitetaan kartoittamalla kyselytutkimuksella toimeksiantajan automaatio-osaston kokemuksia ja mielipiteitä asiakastarpeen määrittelyyn liittyen. Seuraavaksi luodaan helposti ymmärrettävää UML-käyttäytymiskaaviota toimeksiantajan asiakkaan toimintakuvauksen perusteella, sekä luodaan valmiista PLC-projektista UML-rakennekaavio tarkastelua varten. Lopussa on yhteenveto ja pohdinta

1.4 Yritysesittely

Toimeksiantajana tässä opinnäytetyössä toimii Seinäjoella vuonna 2015 perustettu suunnittelutoimisto Caplan Oy (Caplan, i.a.-b). Yritys tarjoaa asiakkailleen mekaniikka- ja automaatiosuunnittelua, projektinhallintaa, teknistä laskentaa sekä Digital Twin-palveluita (Caplan, i.a.-

a). Marraskuussa 2023 yrityksen henkilökunta koostui yrityksen verkkosivun mukaan 22 mekaniikka- ja automaatioalan osaajasta (Caplan, i.a.-b).

Caplanin mekaniikkasuunnittelun palvelut kattavat useat eri teollisuuden alat kuten elintarvike, konepaja, kappaleenkäsittely ja koneenrakennus (Caplan, i.a.-c). Automaatiosuunnittelussa Caplan keskittyy erityisesti kone- ja laitesuunnitteluun, tarjoten tehokkaita ratkaisuja konepaja- ja laitevalmistajille (Caplan, i.a.-d). Kattavaan automaatiosuunnitteluun voidaan sisällyttää tarpeiden mukaan esimerkiksi PLC- ja/tai käyttöliittymäohjelmointia, piirikaavio- ja/tai layout-suunnittelua, sekä konenäköä. Yritys tarjoaa suunnittelun lisäksi myös kaikki edellä mainitut automaatiopalvelut käyttöönottoon asti.

Projektinhallintapalvelu perustuu yrityksen päivittäiseen toimintaan. Projektin vaiheesta tai koosta riippumatta Caplan tarjoaa asiakkailleen osaamistaan (Caplan, i.a.-f). Caplanin tekninen laskenta sisällyttää kattavasti palveluita lujuus- rakenne- ja paineastialaskennasta siilo- ja säiliörakenteisiin varmistaen kestävyyden sekä turvallisuuden (Caplan, i.a.-g).

Digital Twin -palvelun avulla Caplan pystyy luomaan fyysisestä laitteesta virtuaalisen version, jota simuloimalla laitetta voidaan testata jopa ennen fyysistä käyttöönottoa (Caplan, i.a.-h). Näin pystytään optimoimaan laitteen toimintaa, laskemaan kustannuksia ja käyttöönottoaikaa.

2 UML

2.1 Historia ja taustaa

Projektissa monimutkaisuus, ymmärryksen puute ja kommunikaatio-ongelmat luovat haasteita (Holt & Perry, 2010, s. 20). Tähän avuksi on luotu erilaisia mallinnusmenetelmiä, jotka auttavat projektin sidosryhmiä ymmärtämään tarvittavaa tietoa. Kun 1990-luvun loppupuolella haluttiin luoda yksi yhtenäinen mallinnuskieli, ensimmäinen versio Unified Modelling Languagesta syntyi Grady Boochin, Ivar Jacobsonin ja James Rumbaughin toimesta. Se luotiin yhdistämään 1970-, 1980- ja 1990-luvuilla kehitettyjä useita mallinnusmenetelmiä, joiden avulla ei pystytty tehokkaasti viestimään keskenään, sillä eri menetelmien käyttäjän puhuivat eri kielillä (mts.28). UML on kieli ja sitä voidaan käyttää kommunikoimaan mitä tahansa tietoa, eikä sitä tulisi rajoittaa pelkästään ohjelmistoihin. UML on myös ISO-standardoitu vuonna 2005. Viimeisin versio ja standardi tästä mallinnuskielestä, UML 2.5.1, on julkaistu vuonna 2017 (Walker, 2023).

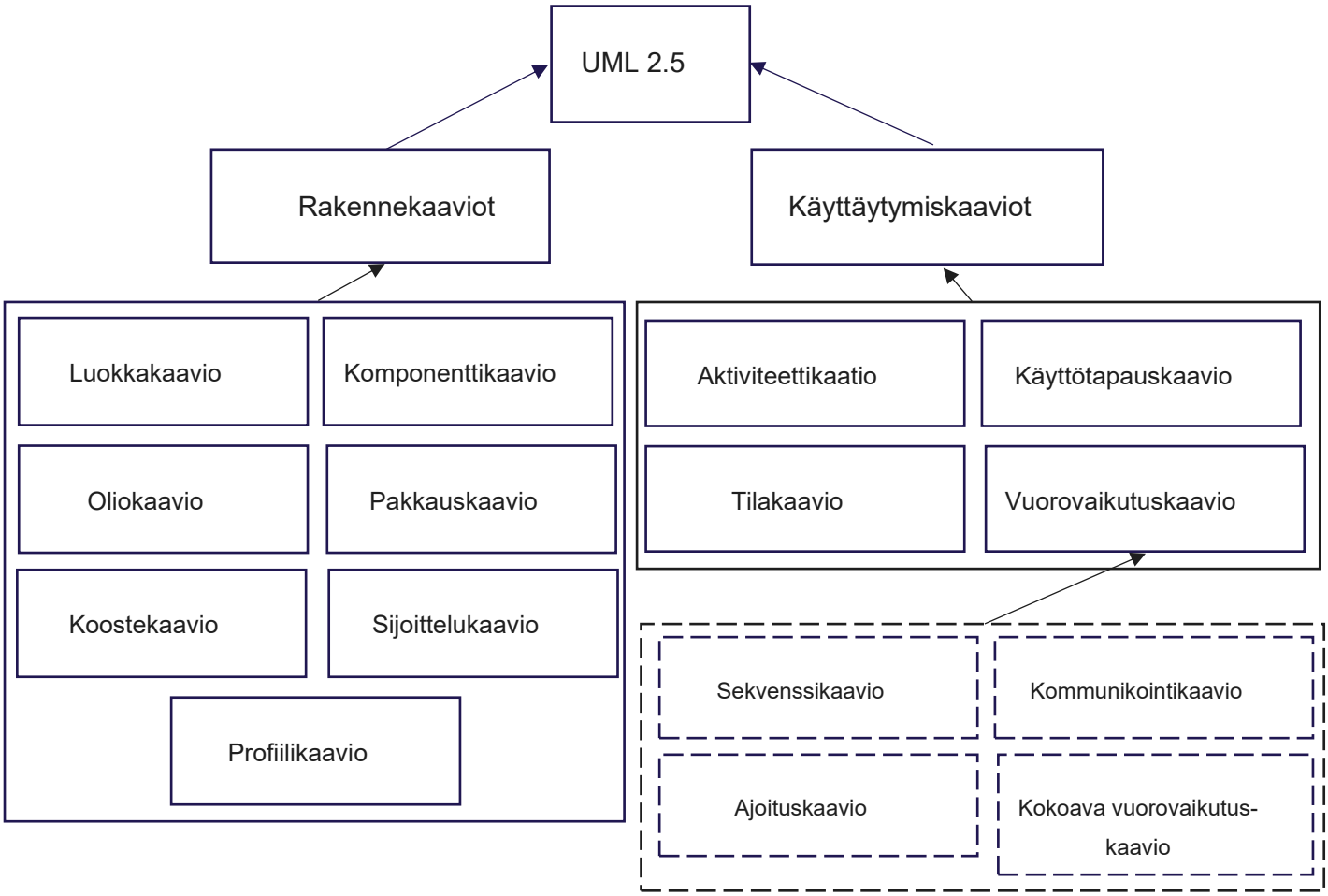
UML menestyi, koska se luotiin vastaamaan teollisuuden tarpeisiin eri elinkaarissa, merkintätavoissa, teollisuudenaloilla ja liiketoiminta-alueilla (Holt & Perry, 2010, s. 28). Sen suosio johtuu myös siitä, että se ei ole yhden organisaation omaisuutta, vaan Object Management Group hallinnoi ja ylläpitää sitä, mikä tekee siitä julkisesti saatavilla olevan standardin.

2.2 Kaaviot

Unified Modelling Languagen pääosassa on UML-kaaviot, joilla on eri käyttökohteet ja käyttötarkoitukset (Walker, 2023). UML-malli koostuu rakenteellisesta sekä käyttäytymisen perspektiivistä (Holt & Perry, 2010, s. 28). Rakenteellinen perspektiivi kuvaa järjestelmän osat ja niiden väliset suhteet, kun taas käyttäytymisen perspektiivi antaa kuvan siitä, miten järjestelmä toimii, mikä on tapahtumien järjestys, millä ehdoilla ne toimivat ja mitä vuorovaikutuksia tapahtumien/asioiden välillä on.

Edellä mainitut rakenteellinen ja käyttäytymisen perspektiivin mukaan UML-kaaviot jaetaan yleisesti kahteen päätasoon, rakennekaavoihin (eng. Structure diagrams) ja käyttäytymiskaavoihin (eng. Behavior diagrams) (Fakhroutdinov, i.a). Päätasoista molemmat sisältävät useita kaaviotyyppejä. Lähteestä riippuen päätasoihin voivat kuulua myös vuorovaikutuskaaviot,

mutta yleisesti ne sisällytetään käyttäytymiskaavoihin, ja niin tässäkin työssä tehdään. Kuvio 1 esittää kuinka UML-kaaviot jakautuvat rakenne- ja käyttäytymiskaavioihin.



Kuvio 1. UML-kaaviorakenne (perustuu Fakhroutdinov, i.a.).

2.2.1 Rakennekaaviot

- Luokkakaavio (eng. Class diagram) kuvaa järjestelmän rakennetta luokkien ja rajapintojen avulla, esittäen myös ominaisuudet, sekä luokkien väliset suhteet (Fakhroutdinov, i.a.). Luokkakaavion avulla on mahdollista luoda yleiskuvaus aiheesta (Gliffy, 2020).
- Oliokaavio (eng. Object diagram) on pitkälti luokkakaavion kaltainen, mutta se kuvaa luokkakaavion yksittäisiä olioita tietyllä hetkellä ja niiden suhteita antaen tarkempaa tietoa oliosta (Gliffy, 2020).

- Komponenttikaavio (eng. Component diagram) luo kuvan osista/komponenteista ja niiden välisistä suhteista halutulla tasolla (Gliffy, 2020).
- Sijoittelukaavio (eng. Deployment diagram) näyttää, kuinka ohjelmiston eri osat liittyvät fyysisiin laitteisiin, jotka suorittavat järjestelmän toimintaa (Walker, 2024).
- Koostekaavio (eng. Composite structure diagram) on lähes kuin luokkakaavio, mutta kertoo luokista ja niiden rakenteista entistä tarkemmin, jolloin koostekaaviota on harvoin tarve käyttää muuhun tarkoitukseen kuin ohjelmistosuunnitteluun (Microsoft 365 Team, 2019).
- Pakkauskaavio (eng. Package diagram) kuvaa ohjelmiston rakennetta ja hierarkiaa pakkausten avulla. Pakkaus voi sisältää esimerkiksi useita kaavioita tai komponentteja (Visual Paradigm, i.a).
- Profiilikaavio (eng. Profile diagram) on kuin UML-kielen erityiskieli, sen avulla mahdollistetaan UML-kaavioiden muokkaaminen eri käyttötarkoituksiin (Microsoft 365 Team, 2019). Profiilikaavioon voidaan määrittää mukautetut stereotyypit, arvot ja rajoitteet.

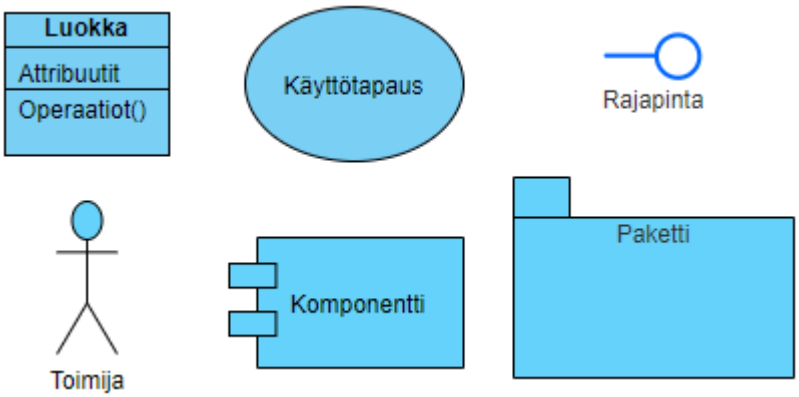
2.2.2 Käyttäytymiskaaviot

- Aktiviteettikaavio (eng. Activity diagram) kertoo kuinka toiminnat ja tapahtumat etenevät järjestelmässä, paljasten miten ohjaus kulkee eri vaiheiden läpi (Gliffy, 2020).
- Käyttötapauskaavio (eng. Use-case diagram) antaa kuvan järjestelmän toiminnasta ilman tarkkaa toteutustapaa (Microsoft 365 Team, 2019). Se esittää tapahtumia, jotka tapahtuvat käyttäjän tai toisen järjestelmän ollessa vuorovaikutuksessa järjestelmän kanssa.
- Tilakaavio (eng. State machine diagram) kuvaa tietyn objektin mahdollisia tiloja ja ehtoja, joiden tulee täyttyä, että objekti pääsee kyseisiin tiloihin (Holt & Perry, 2010, s. 112).
- Vuorovaikutuskaavio (eng. Interaction diagram) jakautuu edelleen neljään alaluokkaan kuten Kuvio 1 osoittaa. Kaaviotyypit ovat ajoitus-, sekvenssi-, kommunikointi- ja kokoava vuorovaikutuskaavio, ja ne esitellään tämän kaaviotyypin jälkeen. Itse vuorovaikutuskaaviota ei erikseen käytetä, vaan alaluokista valitaan parhaiten halutun näkökulman täyttävä kaaviotyyppi (Holt & Perry, 2010, s129).

- Ajoituskaavio (eng. Timing diagram) visualisoi aikajanalla tietyn ajanjakson sisällä tapahtuvaa eri osien toimintaa ja vuorovaikutusta esittäen samalla käsiteltävien objektien päätilat tietyllä hetkellä (Microsoft 365 Team, 2019).
- Sekvenssikaavio (eng. Sequence diagram), esittää kuinka objektit vaihtavat viestejä keskenään ja ovat vuorovaikutuksessa ajan kuluessa (Microsoft 365 Team, 2019).
- Kommunikointikaavio (eng. Communication diagram) on samankaltainen sekvenssikaavion kanssa, mutta tässä kaaviossa pääpainona ovat viestit, joita objektit vaihtavat keskenään, eikä ajan kulumista huomioida (Microsoft 365 Team, 2019).
- Kokoava vuorovaikutuskaavio (eng. Interaction overview diagram) on kuin yleiskatsaus, jonka avulla voidaan yhdistää useita edellä mainittuja vuorovaikutuskaavioita yhdeksi kokonaisuudeksi (Microsoft 365 Team, 2019). Tämä antaa yleiskuvan kaikesta kuvatusta vuorovaikutuksesta.

2.3 Mallinnuselementit ja suhteet

UML-standardissa (Object Management Group (OMG), 2017, s. 683) kerrotaan elementeistä, kuten paketit (packages), luokat (classes) ja käyttötapaukset (use cases), jotka ovat kuin rakennuspalikoita, joista jokainen UML-kaavio muodostuu. Kuvio 2 sisältää yleisiä UML-mallinnuselementtejä.



Kuvio 2. Mallinnuselementtejä.

UML-kaavion tarkoituksena on havainnollistaa järjestelmän toimintaa ja rakennetta, jolloin elementtien välisten suhteiden korostaminen nousee tärkeään rooliin (Nalimov, 2021). Elementtien välisiä suhteita visualisoidaan erilaisilla yhdistävillä viivoilla, nuolilla ja merkinnöillä,

näin lisätään informaatiota kaavioihin selventämällä kaavioiden käyttäytymistä ja niiden yhteyksiä.

2.4 Luokkakaavio

Luokkakaavio luo kuvan järjestelmän rakenteesta hyödyntäen rakenne-elementteinä luokkia ja niiden välisiä suhteita (Fakhroutdinov, i.a.). Luokkakaavion avulla voidaan tarjota yleiskatsaus, miten eri luokat liittyvät toisiinsa, ja missä roolissa ne ovat järjestelmässä.

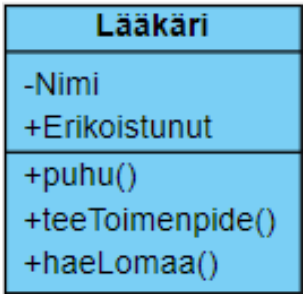
2.4.1 Luokka

Luokkakaavio koostuu pääosin kahdesta pääelementistä, luokista ja suhdeviivoista (Holt & Perry, s. 75). Yksi luokka koostu suorakulmion muotoisesta elementistä, joka on jaettu yleisesti kolmeen osaan, jotka ovat luokka, attribuutti ja operaatio kuten kuviossa 2 on vasemmassa yläkulmassa.

Luokka-osioon määritellään yksinkertaisesti kyseisen luokan nimi, ja tarvittaessa rajoitteita ja lisämääreitä tarkentamaan luokkaa (Järvelä & Puusaari, 2005, s. 18).

Luokan ominaisuuksia ja sisältöä kuvataan attribuuteilla. Attribuuttien näkyvyyttä ja käytettävyyttä muista luokista voidaan rajoittaa (Järvelä & Puusaari, 2005, s. 19). Näkyvyydet jaetaan kolmeen eri tasoon ja näytetään luokassa merkein: julkinen (public) +, yksityinen (private) - ja suojattu (protected) #. Esimerkkinä luokan Potilas yksityisiä attribuutteja voisivat olla -Nimi, -Syntymäpäivä (Unhelkar, 2020, s. 136). Kuviossa 3 on Lääkäri-luokka, jonka attribuutteja ovat Nimi, joka on yksityinen attribuutti, ja Erikoistunut, joka on julkinen attribuutti.

Kolmas osa luokassa on varattu operaatioille (operations/methods), joilla kuvataan luokalle määritetyt tehtävät (Unhelkar, 2020 s. 137). Esimerkkinä luokalla Lääkäri, voisi olla operaatioina puhu(), teeToimenpide() ja haeLomaa() kuten kuviossa 3.

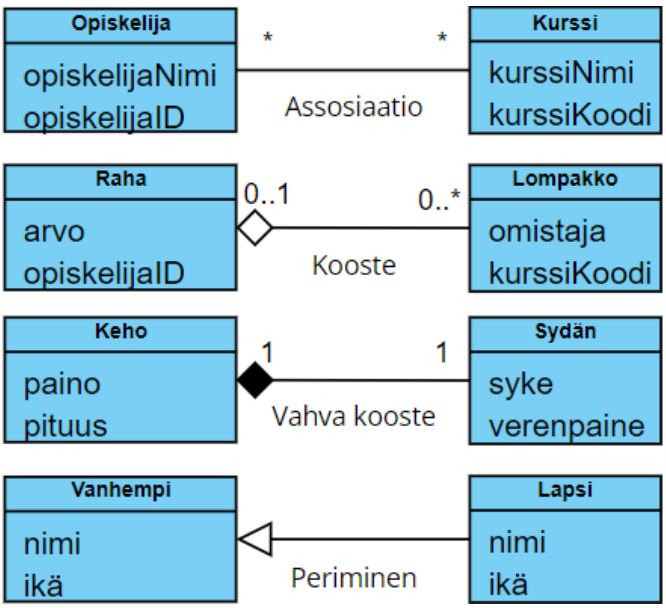


Kuvio 3. Lääkäri-luokka

2.4.2 Suhteet

Luokkakaavion suhteita kuvataan assosiaatiolla, koosteella, vahvalla koosteella ja perimisellä. UML-blogissa (Nalimov, 2021) edellä mainitut suhdeviivat kuvataan näin:

- Assosiaatio (assosiation) kuvaa yksinkertaista vuorovaikutusta. Suhdeviivan ympärille voidaan määritellä tarkempia tietoja suhteesta. Esimerkkinä tästä ovat oppilas ja kurssi, bussi ja matkustaja. Kuviossa 4 esitetään opiskelija- ja kurssi-luokan välistä assosiaatiosuhdetta.
- Koostumussuhde (aggregation) on assosiaation muoto, joka kuvaa elementtien välillä olevan ”on osa” tai ”kuuluu” suhdetta. Esimerkkinä tästä ovat kirjasto ja kirja, raha ja lompakko. Näiden välillä on koostumussuhde, mutta alaluokka ei ole riippuvainen pääluokasta, ja voi olla olemassa myös ilman siihen liitettyä pääluokkaa. Kuviossa 4 esitetään lompakko- ja raha-luokan välistä koostumussuhdetta.
- Vahva koostumussuhde (composition aggregation) on kuin koostumussuhde, mutta eroavaisuus on siinä, että alaluokka on riippuvainen pääluokasta, eli ei voi olla olemassa ilman pääluokkaa. Esimerkkinä tästä ovat takki ja takin tasku, keho ja sydän. Kuviossa 4 esitetään keho- ja sydän-luokan välistä vahvaa koostumussuhdetta.
- Periminen (inheritance) / yleistäminen (generalization) on suhde, jossa elementin ominaisuudet periytyvät toiselle elementille. Esimerkkinä tästä ovat vanhempi ja lapsi, eläin ja kissa. Kuviossa 4 esitetään vanhempi- ja lapsi-luokan välistä perimissuhdetta.



Kuvio 4. Suhdeviivat ja kerrannollisuus

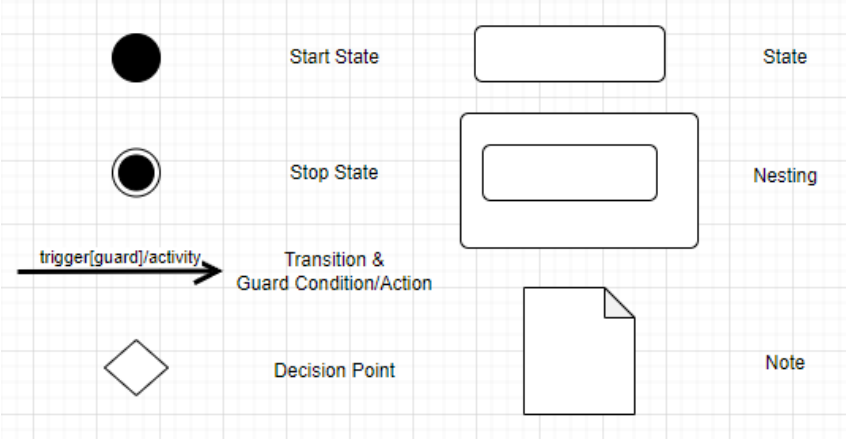
Suhdeviivoihin voidaan lisäksi lisätä kerrannaisuusmerkintöjä (multiplicity), joilla tarkennetaan luokkiin liittyvien kohteiden lukumäärää (Unhelkar, 2020 s. 149). Kuviossa 4 kerrannallisuus-merkintöjä ovat:

- Assosiaatiosuhteessa olevilla Opiskelija- ja Kurssi-luokilla on merkintä *, mikä tarkoittaa, että Opiskelija-luokalla voi olla lukematon määrä Kurssi-luokkia, ja toisinpäin. Tämä kuvastaa sitä, että opiskelija voi osallistua mihin tahansa määrään kursseja, ja samalla kurssi voi olla liitetty mihin tahansa määrään opiskelijoita.
- Koostesuhteessa Raha-luokalla on merkintä 0..1, mikä tarkoittaa, että luokalla voi olla korkeintaan yksi Lompakko-luokka. Toisaalta Lompakko-luokalla voi olla lukematon määrä Raha-luokkia mikä on merkitty 0..*. Tämä tarkoittaa, että jokaisella rahalla voi olla korkeintaan yksi lompakko, mutta lompakossa voi olla useita rahoja.
- Vahvassa koostesuhteessa Keho- ja Sydän-luokan välillä molemmilla on merkintä 1, mikä kertoo, että molemmilla luokilla voi olla vain yksi esiintymä toisen sisällä. Tämä ilmaisee, että jokaisella keholla on tarkalleen yksi sydän, ja jokaisella sydämellä on tarkalleen yksi keho.

2.5 Tilakaavio

Holtin ja Perryn (2010, s. 111–112) mukaan tilakaavio kuvaa objektin käyttäytymistä sen elinkaaren aikana tuoden esiin tapahtumien järjestystä sekä niihin liittyviä ehtoja. Unhelkarin

(2020, s. 237) mukaan tilakaavio rakentamiseen on käytössä kahdeksan piirroselementtiä, jotka esitellään kuviossa 5.



Kuvio 5. Tilakaavion piirroselementit

Unhelkar (2020, s. 237) määrittelee tilakaavion piirroselementit seuraavasti:

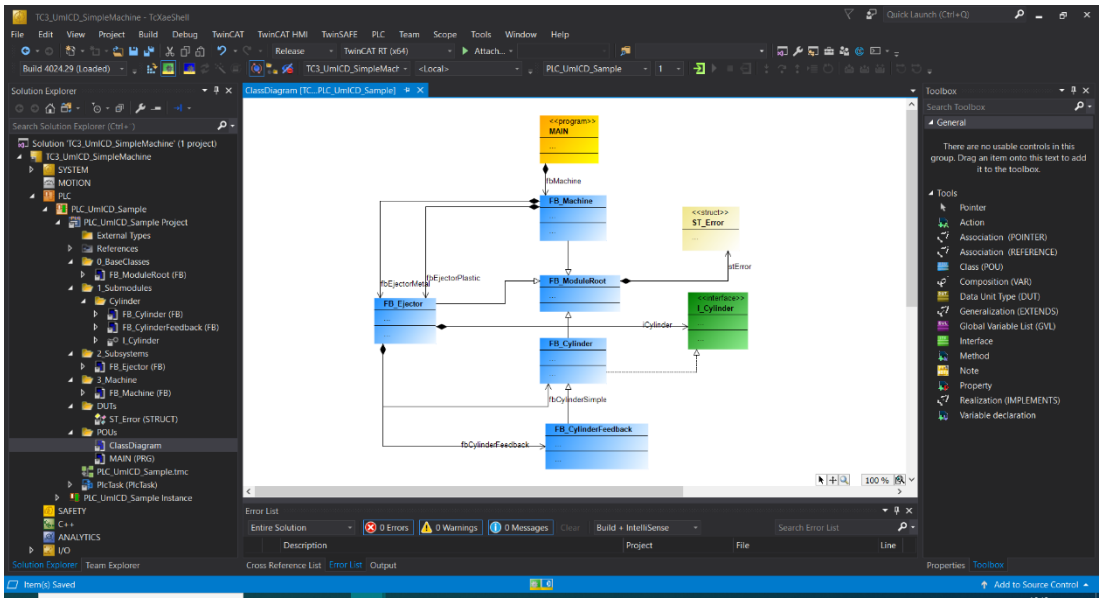
- Aloitustila (Start state) on mallinnuksen aloitusvaiheessa käytettävä pseudotila, jota kuvataan suurella pisteellä. Jokaisessa kaaviossa voi olla vain yksi aloitustila.
- Lopetustila (Stop state) on toinen tilakaaviossa käytettävä pseudotila, jolla kuvataan loppupistettä tai prosessin päättymistä. Lopetustilaa kuvataan tarvittaessa ns. bullseye-symbolilla, mutta lopetustilan olemassaolo ei ole välttämätöntä, toisaalta niiden määrääkään ei ole rajoitettu.
- Siirtymä (Transition) kuvaa muutosta tilasta toiseen nuolella.
- Ehto/Toimenpide (Guard Condition/Action) kuvaa minkä ehdon tulee täyttyä, että siirtymä toteutuu, ja/tai mikä toimenpide toteutetaan siirtymän johdosta. Ehto/toimenpide kuvataan yleisesti siirtymänuolen yläpuolelle.
- Päätöskohta (Decision Point) on kuvattu timanttisymbolina, joka näyttää erilaiset tilavaihtoehdot, joihin objekti voi siirtyä riippuen siitä, mikä määritelty ehto täyttyy.
- Tila (State) kuvaa nimensä mukaan objektin tilaa, ja se on tilakaaviossa keskeisessä roolissa. Tilaa kuvataan pyöristetyllä suorakulmiolla.
- Sisällytys (Nesting) viittaa tiloihin, jotka ovat tilan sisällä. Sitä hyödynnetäänkin usein monimutkaisempien järjestelmien ja prosessien mallintamisessa pienempiin osiin.
- Muistiinpanolla (Note) voidaan lisätä lisätietoa muista piirroselementeistä.

2.6 UML-työkalut

Työn yhtenä tavoitteena oli löytää sopivin UML-työkalu toimeksiantajan käyttöön. Suurimmasta osasta UML-työkaluista löytyy pitkälti samat ominaisuudet, mutta kunkin työkalun eduksi ja haitaksi löytyy myös eroavaisuuksia kokonaisuuksista.

2.6.1 Beckhoff Twincat UML

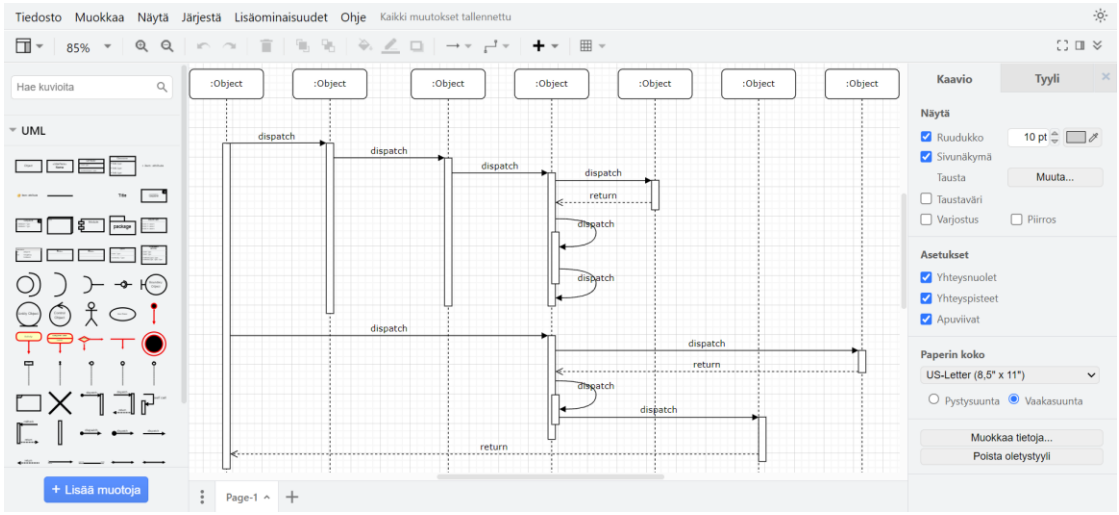
Twincat 3 -ohjelmointiympäristö pitää sisällään TF1910 UML -työkalun, jonka avulla mallinnetusta kaaviosta generoidaan suoraan ohjelmakoodia sekä myös toisinpäin (Beckhoff Automation, i.a.-a). Twincatin UML-työkalulla on kuitenkin mahdollista luoda vain luokka- ja tilakaavioita. Toimeksiantajan ollessa Beckhoff Automationin kumppaniyritys (Caplan, i.a.-e), Twincat on yrityksen PLC-projektien ensisijainen suunnitteluohjelmisto, jolloin muuta UML-työkalua ei tarvittaisi, mikäli UML-tarve ulottuisi vain luokka- ja/tai tilakaaviohin.



Kuvio 6. Beckhoff Twincat 3 -luokkakaavio

2.6.2 Diagrams.net

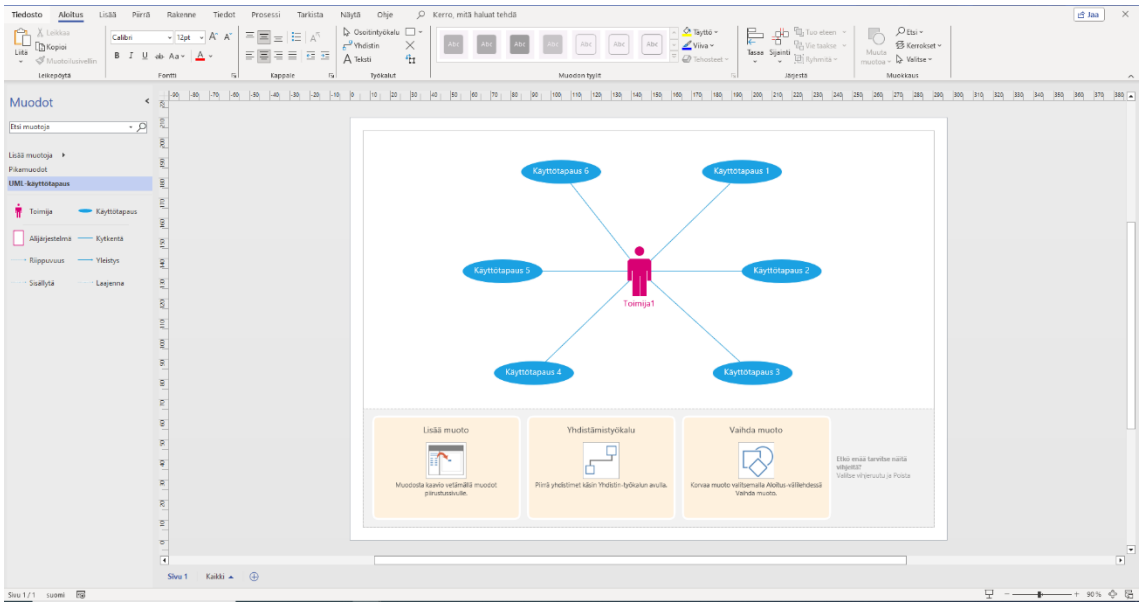
Diagrams.net (ent. Draw.io) on ilmainen kaavioiden piirtämiseen luotu ohjelmisto, joka on saatavilla niin verkossa kuin työpöytäsovelluksena (Draw.io, 2019). Digrams.net ei sisällä kaikista kaaviotyypeistä valmiita pohjia, mutta ohjelman UML 2.5 -muotokirjastosta löytyy laajasti elementtejä ainakin suurinta osaa kaaviotyyppejä varten.



Kuvio 7. Diagrams.net-ohjelmiston sekvenssikaaviopohja

2.6.3 Microsoft Visio

Microsoftin Visio on piirustusohjelma, joka mahdollistaa kaavioiden, kuten UML-kaavioiden luomisen verkko- ja työpöytäsovellusta hyväksi käyttäen (Microsoft, i.a). Visio sisältää valmiita mallipohjia ja muotoja helpottaen eri kaaviotyyppien luomista. UML-mallinnuspohjat kuitenkin vaativat maksullisen tilauksen, kuten työpöytäsovelluskin. Visio ei sisällä kaikkia UML 2.5 -kaaviotyyppejä, vaan valikoima rajautuu niiden osalta luokka-, komponentti-, sijoittelu-, sekvenssi-, aktiviteetti-, tilakone-, käyttötapaus- ja viestintäkaavioihin.



Kuvio 8. Microsoft Visio -käyttötapauskaaviopohja

3 PLC

3.1 Ohjelmoitava logiikka

Ohjelmoitavar logiikkaohjaimet (eng. Programmable Logic Controller), lyhennettynä ja yleisesti tunnettuna PLC:t, ja teollisuus-PC:t ovat vallitsevia ohjauslaitteita teollisuusautomassa (Keinänen & Sumujärvi, 2019, s. 248). Edellä mainitut automaation ohjauslaitteet vastaanottavat anturidataa tulojen kautta ja ohjaavat samalla niihin kytkettyjä toimilaitteita lähtöjen kautta ohjausohjelman mukaisesti. PLC:n ja teollisuus PC:n toiminnat eivät rajoitu ainoastaan anturitiedon käsittelyyn, vaan niitä voidaan hyödyntää laajasti laskennasta säätö- ja valvontatoimintaan, hälytysten käsittelystä raportointiin ja tietoliikennetoimintaan.

PLC:n toiminta perustuu ohjausohjelman määrittämien loogisten toimintojen toteuttamiseen (Keinänen ja Sumujärvi, 2019, s. 254). Logiikka tarkastelee tulojen tilaa muutosten varalta, ja tekee mahdolliset päätökset ja lähtöjen säädöt ohjelman perusteella.

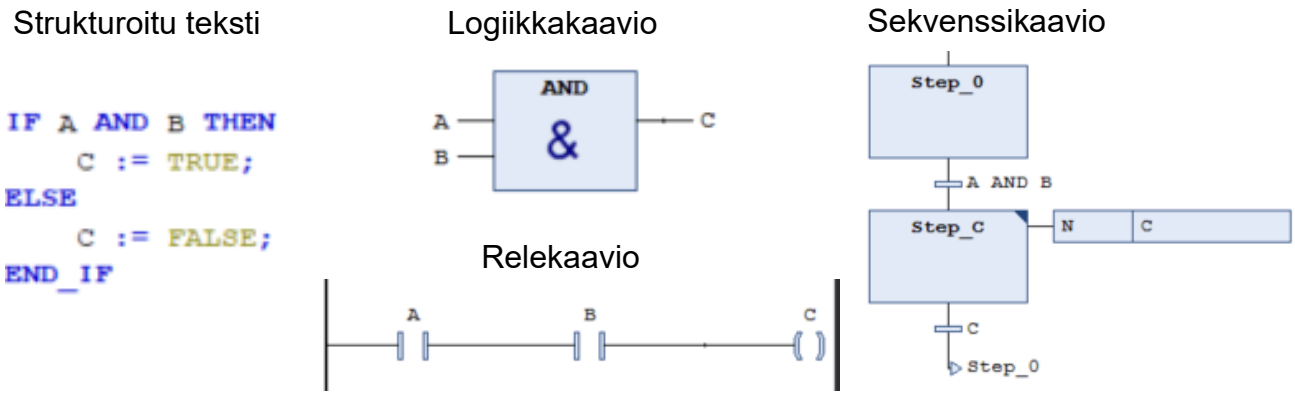
Tulo- ja lähtösignaalit ovat yleisesti joko digitaalisia tai analogisia (Automation Direct, i.a, s. 15). Digitaalisen signaalin mahdolliset arvot ovat on tai off. Lähtönä tällä signaalilla voidaan ohjata esimerkiksi valoja ja releitä päälle/pois. Digitaalinen tulosignaali voisi taas kuvata esimerkiksi, onko tankki täynnä vai ei.

Analoginen signaali voi sijoittua eri aluille, kuten välille 4–20 mA tai 0–10 V (Automation Direct, i.a, s. 16). Analogista signaalia on tarkoituksen mukaista käyttää esimerkiksi kertomaan tulona PLC:lle säiliön nesteen tarkka pinnankorkeus, ja lähtönä säätämään venttiiliä tarkasti tiettyyn asentoon.

3.2 PLC-ohjelmointi

Logiikkaohjaimessa tulee olla ainakin yksi ohjelmoitu ohjelma (Keinänen & Sumujärvi, 2019, s. 259). Tämä ohjelma määrittelee, mitä logiikkaohjain tekee. Järjestelmään määritellään pääohjelma, jota logiikkaohjain suorittaa. Pääohjelma voi myös kutsua muita ohjelmia, toimilohkoja ja funktioita, jolloin niistä tulee osa suoritettavaa ohjelmaa (Keinänen & Sumujärvi, 2019, s. 259).

PLC-ohjelmointi tapahtuu usein IEC 61131-3 -standardin esittämillä ohjelmointimenetelmillä (Keinänen & Sumujärvi, 2019, s. 259). Näitä ovat Instruktiolista (IL), Relekaavio (Ladder, LD), Logiikkakaavio (FBD), Strukturoitu teksti (ST) ja Sekvenssikaavio (SFC). Kaikilla edellä mainituilla ohjelmointikielillä on omat vahvuutensa, kuten visuaalisuus, yksinkertaisuus, käytännöllisyys ja/tai koodin siirrettävyys (Keinänen & Sumujärvi, 2019, s. 259–260). Kuvio 9 esittää osaa IEC 61131-3 -ohjelmointikielistä. Kaikki kuvion esimerkit toteuttavat pitkälti samaa toimintoa: Ehtojen A ja B täyttyessä, C aktivoituu.



Kuvio 9. IEC 61131-3 ohjelmointikieliä (soveltaen Keinänen & Sumujärvi, 2019, s. 261–270)

Ohjelmoinnissa muuttujilla on iso rooli, ohjelmat käsittelevät muuttujia ja niiden arvoja tai sisältöä. PLC-ohjelmoinnissa tyypillisiä muuttujatyyppejä ovat kokonaisluvut (integers), liukuluvut (floating points), aika (time) ja bittimuuttuja (bit strings) (Keinänen & Sumujärvi, 2019, s. 260–261). Näistä pääjaksoista voidaan erotella vielä tarkemmat muuttujatyypit niiden ominaisuuksien perusteella. Taulukko 1 kuvaa joitain esimerkkejä, kuinka pääjaksot jaetaan tarkempiin alajaksoihin esimerkiksi muuttujatyypin käyttötarkoituksen ja/tai alueen perusteella.

Kokonaisluvut	INT	-32768-32767
	USINT	0-255
Liukuluku	REAL	±10^±38
	LREAL	±10^±308
Aika	TIME	T#10d4h38m57s12ms
	DATE	D#1989-05-22
Bittimuuttuja	BOOL	1 bit
	BYTE	8 bits

Taulukko 1. Muuttujatyyppejä (perustuu Keinänen & Sumujärvi, 2019, s. 260–261)

Toimilohko, toisin sanoin myös Funktion Block (FB), on ohjelmoinnissa monikäyttöinen ja kopioitavissa oleva osa, joka mahdollistaa myös muiden toimilohkojen ja funktioiden kutsumisen ja käyttämisen (Keinänen & Sumujärvi, 2019, s. 260). Funktion Blockilla on oma sisäinen muisti, jonka avulla toimilohkon omien muuttujien arvo voidaan säilyttää seuraavaan suorituskertaan.

Funktiolla ei toimilohkosta poiketen ole omaa sisäistä muistia, sitä käytetäänkin usein vain suorittamaan laskutoimituksia ja muita toistuvia tehtäviä (Keinänen & Sumujärvi, 2019, s. 260). Funktion suoritettua toimintansa, sen omat muuttujat palautetaan oletusarvoihin odottamaan seuraavaa suorituskertaa.

4 Asiakastarpeen määrittämisen haasteet

4.1 Kyselytutkimus

Ensimmäinen työn kahdesta päätavoitteesta oli tehostaa toimeksiantajan automaatioprojektien asiakastarpeiden määrittelyä UML-kaavioita hyväksi käyttäen. Jotta voidaan löytää tarkoituksenmukainen UML-kaavio, meidän on ensin tunnistettava ongelmat ja haasteet.

Haasteiden tunnistamiseksi, suoritettiin toimeksiantajan automaatio-osastolla kysely, jonka avulla perehdyttiin siihen, millaisia haasteita epäselvä asiakastarpeen määrittely aiheuttaa, ja mitä näistä haasteita seuraa. Kysely lähetettiin viidelle henkilölle, joista kaikki vastasivat.

Kysely suoritettiin Google Forms -alustalla ja sen kysymykset keskittyivät automaatio-osaston kokemuksiin projektien asiakastarpeiden määrittelystä ja sen vaikutuksista työhön. Kysymykset käsittelivät muun muassa epäselvän asiakastarpeen yleisyyttä, sen aiheuttamia ongelmia, mahdollisia vaikutuksia työaikaan, työn mielekkyyteen, työtehoon ja stressitasoon, sekä keinoja minimoida asiakastarpeen epäselvä määrittely.

4.2 Kyselyn tulosten yhteenveto

Kyselyn vastauksista ilmeni, että projekteissa koetaan usein tai jopa aina puutteita asiakastarpeen määrittelystä. Vastaajien mukaan epäselvät asiakastarpeen määrittelyt aiheuttavat merkittäviä ongelmia, kuten projektin aikataulun venymistä, lisätöitä, muutoksia ja laadun heikkenemistä. Kyselyn vastauksista käy esille, että asiakastarpeen epäonnistunut määrittäminen koetaan johtuvan usein siitä, ettei asiakas tiedä oikein itsekkään mitä projektilta halutaan. Tällöin esimerkiksi automaatioprojektin toimintakuvauksesta tulee puutteellinen. Tästä johtuen asiakkaalle ei välttämättä osata esittää oikeita kysymyksiä esimerkiksi laitehankinnan osalta, jolloin myös projektin kustannusarviot saattavat heitellä merkittävästi. Lisäksi tilanne vaikuttaa vastaajien työskentelykykyyn stressin noustessa ja työn mielekkyyden laskiessa.

Ratkaisuna ja/tai helpotuksena epäonnistuneen asiakastarpeen määrittelyyn vastauksissa ehdotetaan useita käytännön asioita. Vastauksista käy esille tarvetta entistä parempaan kommunikointiin asiakkaan kanssa esimerkiksi pyytämällä parempia toimintakuvauksia selkeyttämään tarpeita projektin varhaisessa vaiheessa. Tämän tueksi esitettiin tarve kehittää

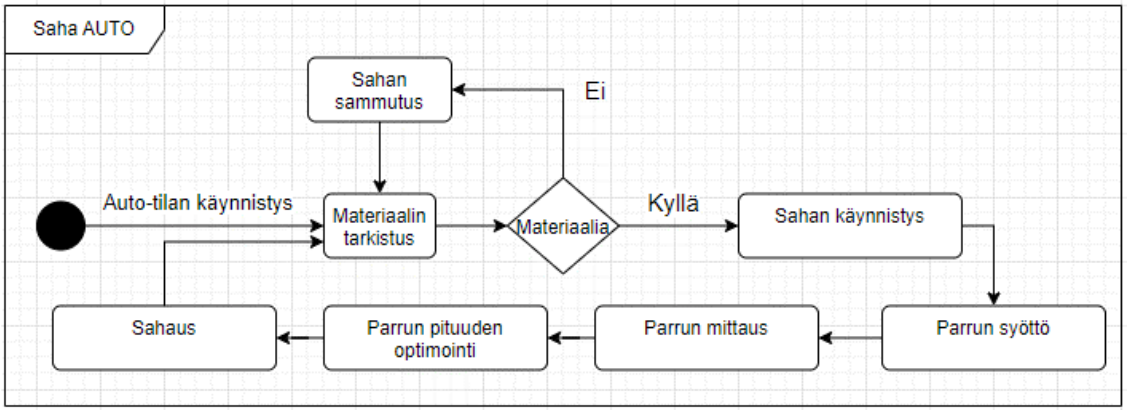
menetelmiä tai työkaluja, joiden avulla asiakkaan tarpeet saataisiin paremmin ja selkeämmin huomioitua, jolloin projektin toteuttaminen olisi tehokkaampaa niin tekijöille kuin asiakkaallekin.

5 Asiakastarpeen määrittäminen UML-mallinnuksella

Työn toisena päätavoitteena oli tutkia, kuinka UML-mallinnusta voitaisiin hyödyntää asiakastarpeen määrittelyssä. Tässä osiossa luodaan konkreettinen esimerkki siitä, kuinka asiakkaan luoma toimintakuvaus voidaan visualisoida hyväksikäyttäen UML-mallinnuskieltä Diagrams.net-mallinnustyökalulla. Toimintakuvaus mallinnetaan UML -tilakaavioksi, jossa tiloja esittävät toimintakuvauksen päätapahtumat siten, että kaavion lukeminen sekä ymmärtäminen ei vaadi teknistä asiantuntemusta.

Mallinnettu kone on jaettu kolmeen osaan, saha, naulain ja pinkkari. Kustakin koneenosasta on saatu asiakkaalta hyvin yleistasoinen toimintakuvaus, joiden pohjalta laaditaan tilakaaviot, jotka mahdollistavat nopealla silmäyksellä yleiskuvauksen koneen toiminnasta. Luotujen tilakaavioiden ei ole tarkoitus olla yksityiskohtaisia, vaan antaa lukijalle peruskäsitys koneenosien toiminnasta, eivätkä ne ota kantaa mahdollisiin häiriötilanteisiin.

Tämän luvun UML-tilakaaviot on mallinnettu Diagrams.net-mallinnustyökalua käyttäen.



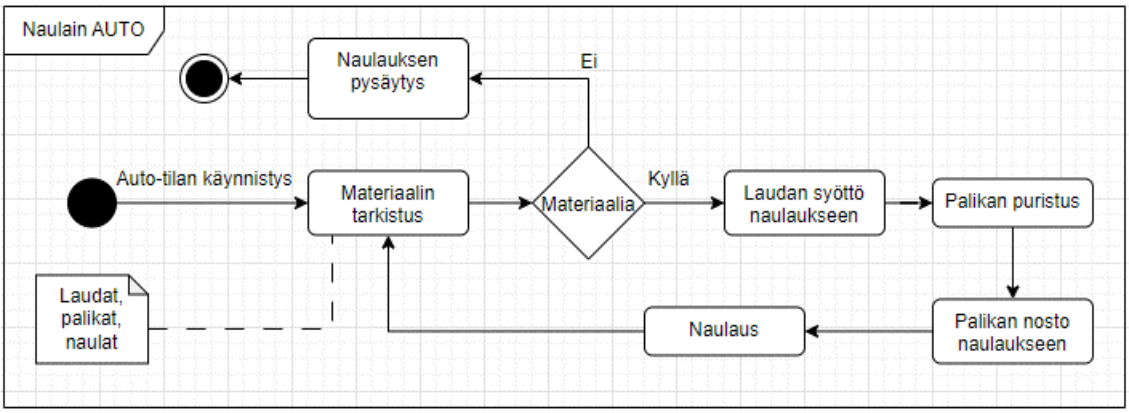
Kuvio 10. Saha-tilakaavio

Ensimmäinen tilakaavio luotiin sahan perustoimintaperiaatteesta. Sahalla on tiloja yhteensä seitsemän. Ehtona etenemiselle seuraavaan tilaan on, että sen hetkisen tilan tavoite saavutetaan. Kuten kuvio 10 esittää, Automaatti-tilan käynnistämisen jälkeen tilat suoritetaan järjestyksessä olettaen, ettei materiaalia puutu:

1. Materiaalin tarkistus.
   Päätöskohta: Ei materiaalia → Sahan sammutus -tila, materiaalia → Sahan käynnistys -tila.

2. Sahan käynnistys.
3. Parrunsyöttö.
4. Parrun mittaus.
5. Parrun pituuden optimointi.
6. Sahaus.

Viimeisen tilan jälkeen palataan takaisin materiaalin tarkistus -tilaan ja kierto alkaa alusta.

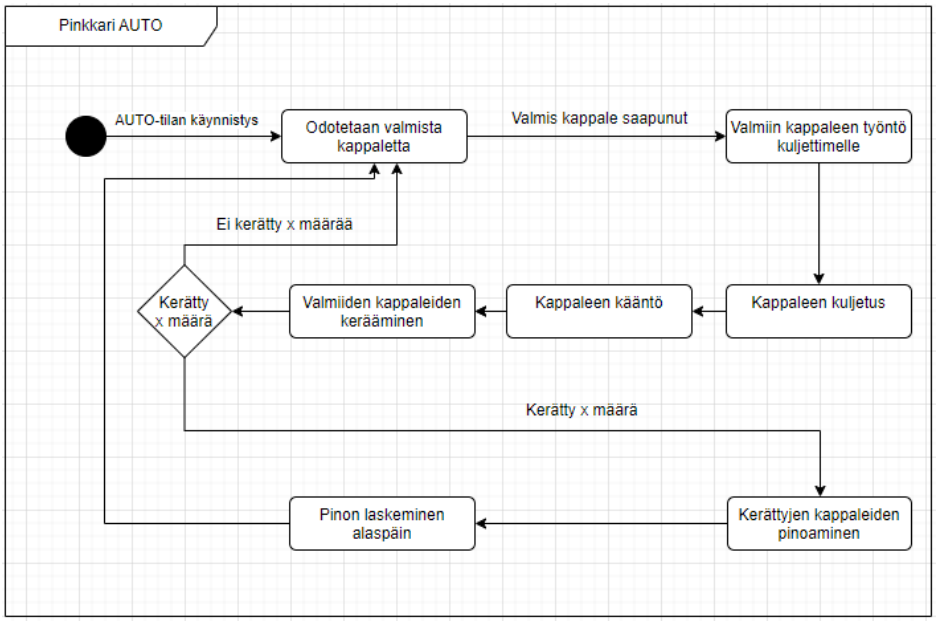


Kuvio 11. Naulain-tilakaavio

Toinen tilakaavio luotiin naulaimen perustoimintaperiaatteesta. Naulaimella on tiloja kuusi kappaletta. Ehtona etenemiselle on sen hetkisen tilan tavoitteen täyttyminen ennen etenemistä seuraavaan tilaan. Kuten kuvio 11 esittää, automaattitilan käynnistyksen jälkeen tilat suorittaan järjestyksessä olettaen, ettei materiaalia puutu:

1. Materiaalin tarkistus.
   Päätöskohta: Ei materiaalia → Naulauksen pysäytys -tila, Materiaalia → Laudan syöttö naulaukseen -tila.
2. Laudan syöttö naulaukseen.
3. Palikan puristus.
4. Palikan nosto naulaukseen.
5. Naulaus.

Viimeisen tilan jälkeen palataan takaisin materiaalin tarkistus -tilaan ja kierto alkaa alusta.



Kuvio 12. Pinkkari-tilakaavio

Kolmas ja viimeinen tilakaavio tehtiin pinkkari-koneenosasta. Pinkkarin tilakaaviossa on seitsemän tilaa. Kuten edellisten koneenosien tilakaaviossa, on etenemisen ehtona sen hetkisen tilan tavoitteen täyttyminen. Koneenosan toimintoa suoritetaan kuvion 12 mukaisesti järjestyksessä automaattitilan käynnistämisen jälkeen olettaen, että kierto etenee suunnitellusti:

1. Odotetaan valmista kappaletta.
2. Valmiin kappaleen työntö kuljettimelle.
3. Kappaleen kuljetus.
4. Kappaleen kääntö.
5. Valmiiden kappaleiden kerääminen.
   Päätöskohta: Ei kerätty x määrää → Odotetaan valmista kappaletta -tila, kerätty x määrä →Kerättyjen kappaleiden pinoaminen -tila.
6. Kerättyjen kappaleiden pinoaminen.
7. Pinon laskeminen alaspäin.

Viimeisen tilan jälkeen palataan takaisin odotetaan valmista kappaletta -tilaan ja kierto alkaa alusta.

6 PLC-projektin mallintaminen UML-kaaviolla

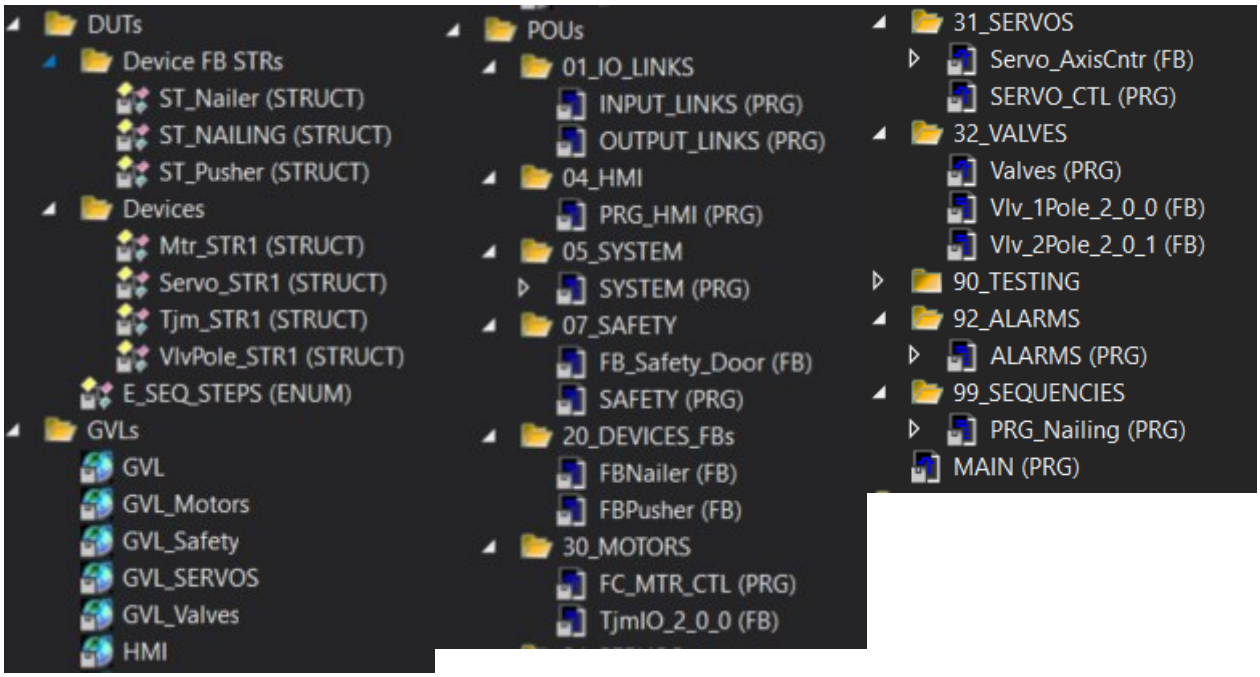
6.1 Luokkakaavion luominen Beckhoff Twincatilla

Tässä osiossa tutkitaan, kuinka UML-mallinnusta hyödyntäen luodaan PLC-ohjelmasta sen rakennetta kuvaava kaavio projektin ylläpidettävyyttä ja dokumentointia varten, joka on työn toinen keskeinen tavoite. Tämän avulla projektikoodin sisältöön ja rakenteeseen päästäisiin uudelleen kiinni nopeasti projektin jälkeen. Lisäksi projektin aikana myös uudet tiimin jäsenet pääsevät nopeasti selville projektikoodin sisällöstä ja rakenteesta.

Unified Modelling Languagen ja PLC-koodin yhdistäminen luokkakaavioksi tuo useita etuja etenkin monimutkaisten järjestelmien kehittämiseen ja ylläpitoon, sekä dokumentointiin, analysointiin ja laajentamiseen (Beckhoff Automation, i.a.-b).

UML-luokkakaavio PLC-ohjelmasta luotiin käyttäen työkaluna Beckhoff Twincat 3 -ohjelmointiympäristön UML-työkalua. Työkalulla on mahdollista generoida suoraa ohjelmakoodista UML-luokkakaavio, ja sitä hyödynnettiin kaavion luomisessa. Jotain kansiorakenteen tiedostojen nimiä on muokattu yleiskielisemmiksi salassapidon vuoksi. Kyseessä on eri projekti kuin osion 5 mallinnuksessa.

Mallinuksen ensimmäinen vaihe oli määrittää PLC-ohjelman kansiorakenteesta, mitä osia sisällytetään luokkakaavioon.



Kuvio 13. PLC-ohjelman kansiorakenne

Ohjelma sisältää monta ohjelmaa (PRG) kuten kuvio 13 esittää. Pääohjelmaksi on määritelty MAIN-ohjelma. Beckhoff on määrittänyt omassa UML-työkalussaan synonyymit UML-luokkakaavion elementeille, jolloin UML saadaan entistä paremmin soveltumaan myös PLC-ohjelmointiin. Taulukko 2 esittää nämä synonyymit, niiden avulla voidaan luoda luokkakaavio PLC-ohjelmasta, ja ymmärtää sen sisältöä paremmin.

Taulukko 2. UML-synonyymit IEC 61131-3 (perustuu Beckhoff automation, i.a.-b).

Termit olio -ohjelmoinnissa	Synonyymi IEC 61131-3
Luokka (UML: class)	POU-tyypit Ohjelma (PRG): PROGRAM Toimilohko (FB): FUNCTION_BLOCK Funktio (FUN): FUNCTION
Attribuutti (UML: attribute) Sisäinen muuttuja Parametri: {input} Ominaisuus: {property} Lähtevä parametri: {output}	Muuttujatyypit Muuttujat: VAR Tulevat muuttujat: VAR_INPUT Ominaisuus: PROPERTY Lähtevät muuttujat: VAR_OUTPUT
Operaatio (UML: operation)	Metodi: METHOD Toiminta
Rajapinta (UML: interface)	Rajapinta: INTERFACE
	Globaali muuttujalista (GVL): VAR_GLOBAL
	Käyttäjän määrittelemä datatyyppi (DUT): TYPE

Liitteessä 1 on Beckhoff Twincatin generoima luokkakaavio kuvion 13 PLC-projektista. Luokkakaavio on siistitty helpommin ymmärrettävään muotoon poistamalla funktio-luokat, joihin ei ole varsinaisia suhdeviivoja, sillä niitä ei määritellä näytettävissä luokissa, vaan niitä kutsutaan suoritettavista ohjelmista. Lisäksi generoidusta luokkakaaviosta saatiin vielä helpommin ymmärrettävä järjestelemällä luokat siten, etteivät suhdeviivat ylitä toisiaan.

6.2 Luokkakaavion tulkinta

UML-kaavioon otettiin tässä työssä mukaan ne kansiorakenteen ohjelmat, toimilohkot ja globaalit muuttujalistat, jotka ovat ohjelman toiminnallisuuden kannalta välttämättömiä. Luokkien attribuutteja eli muuttujia eikä operaatioita eli metodeja/toimintoja ei näytetä mallinnetun projektinkoodin ollessa toimeksiantajan omaisuutta.

Projekti sisältää toimilaitteina 11 venttiiliä sekä kaksi moottoria, joita ohjaamalla projektin laite saadaan käyttäytymään halutulla tavalla.

Liitteessä 1 kultaiset luokat ovat ohjelmia (PRG), siniset toimilohkoja (FB), keltaiset itse luotuja datatyyppejä (DUT) ja vaaleanpunaiset globaaleja muuttujalistoja (GVL).

- PRG_HMI-, ALARMS- ja MAIN-ohjelmilla ei ole suhteita muihin luokkiin. Esimerkiksi MAIN-ohjelmassa kyllä kutsutaan muita ohjelmia koko ohjelman suorittamisen mahdollistamiseksi, mutta MAIN-ohjelmassa ei määritellä niitä.

- SYSTEM-ohjelmalla on vahva kompositiosuhde E_SEQ_STEPS-struct-muuttujatyyppiin, joka on määritetty muuttujalla NSTEP.

- SAFETY-ohjelmalla on vahva kompositiosuhde toimilohkoon FB_Safety_Door muuttujalla FBSafetydoor.

- FC_MTR_CTL-ohjelmalla on vahva kompositiosuhde TjmIO_2_0_0-toimilohkoon, joka on määritetty muuttujalla M_1M2_CTL. TjmIO_2_0_0-toimilohkoon on liitetty Tjm_STR1-datatyyppiä oleva muuttuja STR_M_1M2, jolloin Tjm_STR1-datatyypilla on vahvat kompositiosuhteet sekä toimilohkoon TjmIO_2_0_0 että GVL_Motors-muuttujalistaan, johon se on määritelty.

- SERVO_CTL-ohjelmalla on vahva kompositiosuhde Servo_AxisCntr-toimilohkoon muuttujalla M_1M1_CTL. Servo_AxisCntr-toimilohkoon on liitetty Servo_STR1-datatyyppiä oleva muuttuja M_1M1_CTL_STR, jolloin Servo_STR1-datatyypilla on vahvat kompositiosuhteet sekä toimilohkoon Servo_AxisCntr että GVL_SERVOS-muuttujalistaan, johon se on määritelty.

- PRG_Nailing-ohjelmalla on useita vahvoja kompositiosuhteita muihin luokkiin.
  - stNailing-muuttujan kautta ST_NAILING-datatyyppiin.
  - stNailer1-, stNailer2- sekä stNailer3-muuttujien kautta ST_Nailer-datatyyppiin.
  - FBNailer1-, FBNailer2- sekä FBNailer3-muuttujien kautta FBNailer-toimilohkoon.
  - FBNailer-toimilohkossa on lisäksi määritelty ST_Nailer-datatyypiksi stNailer-muuttuja, jolloin oheisen toimilohkon ja datatyypin välillä on myös vahva kompositiosuhde.
  - FB_BoardPusher-muuttujan kautta FBPusher-toimilohkoon
  - FBPusher-toimilohkossa on lisäksi määritelty ST_Pusher-datatyypiksi stPusher-muuttuja, jolloin oheisen toimilohkon ja datatyypin välillä on myös vahva kompositiosuhde.
  - stPusher-muuttujan kautta ST_Pusher-datatyyppiin.

- Valves-ohjelmalla on myös useita vahvoja kompositiosuhteita muihin luokkiin.
  - Muuttujien Valve_1V3-Valve 1V12 kautta toimilohkoon Vlv_1Pole_2_0_0.
  - Vlv_1Pole_2_0_0-toimilohkoon on liitetty io_STR_CNT-muuttujaan VlvPole_STR1-datatyyppiä oleva muuttuja, joka on määritelty muuttujalistassa GVL_Valves.
  - Muuttujan Valve_1V13 kautta toimilohkoon Vlv_2Pole_2_0_1.
  - Vlv_2Pole_2_0_1-toimilohkoon on liitetty io_STR_CNT-muuttujaan VlvPole_STR1-datatyyppiä oleva muuttuja, joka on määritelty muuttujalistassa GVL_Valves.

Lueteltuna luokat ja suhteet voivat olla epäselvää luettavaa, mutta itse kaavio (liite 1) antaa nopeasti yleiskuvauksen projektiin etenkin henkilölle, jolla on jo kokemusta PLC-ohjelmoinnista. Tämä korostaa sitä millainen apu UML-kaaviosta on monimutkaisen asian ymmärtämisessä.

7 Yhteenveto ja pohdinta

Tässä osiossa käydään läpi työn yhteenveto, sekä pohditaan kuinka UML-mallinnusta voisi jalostaa tulevaisuudessa eteenpäin PLC-ohjelmoinnissa.

7.1 Yhteenveto ja johtopäätökset

Tämän opinnäytetyön päätavoitteena oli tarkastella UML-mallinnuksen hyödyntämistä asiakastarpeen määrittelyssä ja PLC-ohjelmoinnissa. Työ eteni aloittaen ensin katsauksella yleisesti UML-mallinnukseen, sekä perehtymällä sen jälkeen paremmin kahteen UML-kaaviotyyppiin. Seuraavaksi työssä käytiin läpi, mitä tarkoitetaan PLC:llä ja sen ohjelmoinnilla jatkaen työn tutkimusosuuteen, joka aloitettiin kyselytutkimuksella asiakastarpeen määrittelyn haasteista. Työn tutkimusosuus jatkui teoriaosuudessa tarkemmin käsiteltyjen UML-kaaviotyyppien soveltamisella käytännössä.

Työn keskeinen tavoite oli asiakastarpeen määrittelyn haasteiden tunnistaminen, ja sen vaikutukset toimeksiantajan automaatioprojektien etenemiseen. Kyselytutkimuksen avulla saatiin arvokasta toimeksiantajan automaatio-osaston kokemuksista asiakastarpeiden määrittelyyn liittyen. Kyselyn vastausten perusteella pystyttiin luomaan selkeämpi käsitys siitä, miten UML-mallinnus voisi toimia apuna asiakastarpeiden määrittelyssä.

Kyselyn jälkeen asiakastarpeen kartoittaminen siirtyi konkreettisempaan vaiheeseen UML-tilakaavioiden avulla. Luodut tilakaaviot loivat visuaalisen esityksen koneen toiminnasta yleisellä tasolla, perustuen asiakkaan antamaan toimintakuvaukseen.

Lopuksi valmiin automaatioprojektin PLC-ohjelmasta luotiin luokkakaavio, joka auttaa perehtymään nopeasti projektin ohjelmakoodin perusrakenteeseen. Luokkakaavion avulla voitiin hahmottaa miten ohjelmakoodin eri osat, eli kaaviossa luokat, liittyvät toisiinsa. Tämä on tärkeää projektin ylläpidon ja jatkokehityksen kannalta. Lisäksi projektin luokkakaavion avulla mahdolliset uudet tiimijäsenet pystyvät perehtymään projektiin ja sen toteutukseen helpommin.

Yhteenvetona voidaan todeta, että UML-mallinnus tarjoaa hyviä työkaluja asiakastarpeen määrittelyssä ja PLC-ohjelmoinnissa. Visuaaliset mallit, kuten tilakaaviot ja luokkakaaviot, helpottavat monimutkaisten järjestelmien suunnittelua ja ymmärtämistä.

7.2 Pohdinta

Työssä tutkittiin, kuinka UML-mallinnusta voitaisiin hyödyntää asiakastarpeen määrittelyssä sekä PLC-ohjelman dokumentoinnissa ja ylläpidettävyydessä. Tulevaisuudessa olisi mielenkiintoista sekä tarkoituksen mukaista jalostaa osiossa 5 luoduista yleiskuvaavasta tilakaavioista yksityiskohtaisempia UML-kaavioita tarkemmalla tasolla. Kyseiset tilakaaviot luotiin mahdollisimman helposti ymmärrettäviksi, jolloin myös henkilö, joka ei ole tekniikan asiantuntija, pystyisi kaavioita lukemaan. Osion 5 tilakaavioista myös suunnittelija saa nopeasti yleiskäsityksen tarvittavan ohjelman perusrakenteesta koneen ohjaamiseen, sekä ne voivat auttaa suunnittelijaa hahmottamaan mahdolliset ongelmakohdat ja pullonkaulat, mutta osion 5 tilakaavioiden perusteella itse ohjelmointia voi olla vaikea suorittaa kovinkaan pitkälle.

Tilakaavion jalostaminen esimerkiksi sekvenssikaavioksi tulisi tutkia tarkemmin tulevaisuudessa, jolloin itse kaavioon voitaisiin sisällyttää toimilaitteita yksityiskohtaisemmin. Sekvenssikaavion pohjalta voisi jopa ohjelmointi onnistua suunnittelijalta, kun pystyttäisiin jo kaaviossa yksityiskohtaisesti määrittämään, miten ja milloin ohjelmoitava logiikka kommunikoi toimilaitteiden, kuten moottorien, venttiilien ja antureiden kanssa.

Lisäksi myös luokkakaavion soveltamista jo projektin suunnitteluvaiheessa voitaisiin tutkia. Tässä työssä valmiista projektista luotiin luokkakaavio, mutta luokkakaavion avulla voitaisiin myös ennen projektia suunnitella projektin rakennetta, ja kuinka ohjelmat sekä koodi jaettaisiin, jotta projektista tulisi mahdollisimman selkeä.

LÄHTEET

Automation Direct. (i.a). *Practical Guide to Programmable Logic Controllers.* https://instrumentationtools.com/wp-content/uploads/PDF/Books/PLC%20Handbook.pdf

Beckhoff Automation. (i.a.-a). *TF1910 TwinCAT 3 UML.* https://document.beckhoff.com/tf1910.pdf?target=tf1910&lang=en-en

Beckhoff Automation. (i.a.-b). *TF1910 TwinCAT 3 UML. Basic principles.* *https://infosys.beckhoff.com/english.php?content=../content/1033/tf1910_tc3_uml/1510519307.html&id=*

Caplan. (i.a.-a). *Palvelut. https://www.caplan.fi/palvelut/*

Caplan. (i.a.-b). Caplan. https://www.caplan.fi/caplan/

Caplan. (i.a.-c). Mekaniikkasuunnittelu. https://www.caplan.fi/palvelut/mekaniikkasuunnittelu/

Caplan. (i.a.-d). Automaatiosuunnittelu. https://www.caplan.fi/palvelut/automaatiosuunnittelu/

Caplan. (i.a.-e). *PLC-ohjelmointi.* https://www.caplan.fi/palvelut/automaatiosuunnittelu/plc-ohjelmointi/

Caplan. (i.a.-f). *Projektinhallinta.* https://www.caplan.fi/palvelut/projektinhoito/

Caplan. (i.a.-g). *Tekninen laskenta. https://www.caplan.fi/palvelut/tekninenlaskenta/*

Caplan. (i.a.-h). *Digital Twin – Virtuaalinen testaus. https://www.caplan.fi/palvelut/digitaaliset-palvelut/digital-twin-virtuaalinen-testaus/*

Draw.io (21.1.2019). *UML 2.5 shape library with updated shapes*. *Draw.io.* https://www.drawio.com/blog/uml-2-5

Fakhroutdinov, K. (i.a). *The Unified Modeling Language: UML 2.5 Diagrams Overview.* uml-diagrams.org. Haettu 9.12.2023, https://www.uml-diagrams.org/uml-25-diagrams.html

Gliffy. (20.1.2020). *UML Diagrams: 14 Types and Templates. https://www.gliffy.com/blog/uml-2-5-diagram-types-and-templates#uml-diagram-types*

Holt, J., & Perry, S. (2010). *Modelling Enterprise Architectures*. Institution of Engineering and Technology (The IET).

Järvelä, E., & Puusaari, E. (2005). *UML-käsikirja.* Jyväskylän yliopisto, Chydenius-instituutti-Kokkolan yliopistokeskus. https://jyx.jyu.fi/bitstream/handle/123456789/18019/951-39-2153-0.pdf

Keinänen, T., & Sumujärvi, M. (2019). *Automaatiotekniikka*. Sanoma Pro.

Microsoft 365 Team. (24.9.2019). *Pikaopas UML-kaavioihin ja tietokantojen mallinnukseen.* *https://www.microsoft.com/fi-fi/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling*

Microsoft. (i.a). *UML diagrams in Visio.* https://support.microsoft.com/en-au/office/uml-diagrams-in-visio-ca4e3ae9-d413-4c94-8a7a-38dac30cbed6#OfficeVersion=Web

Nalimov, C. (7.6.2021). UML relationships explained: Dependency, Realization, Association, and more. Gleek. *https://www.gleek.io/blog/uml-relationships*

Object Management Group (OMG). (2017). *Unified Modeling Language: Version 2.5.1 (ISO/IEC 19505-2:2012).* https://www.omg.org/spec/UML/2.5.1/PDF

Unhelkar, B. (2020). *Sofware Engineering with UML.* CRC Press.

Visual Paradigm. (i.a.). *What is Package Diagram?*. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-package-diagram/

Walker, A. (21.11.2023). *UML Diagrams: History, Types, Characteristics, Versions, Tools.* GURU99. https://www.guru99.com/uml-diagrams.html

Walker, A. (24.2.2024). *Deployment Diagram: UML Tutorial with EXAMPLE.* GURU99. https://www.guru99.com/deployment-diagram-uml-example.html

LIITTEET

Liite 1. UML luokkakaavio

Liite 1. UML luokkakaavio