TAMPEREEN TEKNILLINEN YLIOPISTO
Tuotantotekniikan laitos

SAMPO IMMONEN

PAPERIKONEEN LINJAKÄYTTÖJEN TEHONTARVELASKENNAN LINJATASON INTEGRAATIO

Diplomityö

Tarkastaja: prof. Asko Riitahuhta
Tarkastaja ja aihe hyväksytty
6.kesäkuuta 2012

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tuotantotekniikan koulutusohjelma
IMMONEN, SAMPO: Paperikoneen linjakäyttöjen tehontarvelaskennan linjatason integraatio
Diplomityö, 92 sivua, 0 liitesivua
Lokakuu 2012
Pääaine: Tuotekehitys
Tarkastaja: professori Asko Riitahuhta
Työn ohjaaja (Metso Oyj): Antti Ketolainen
Rahoittaja: Metso Oyj (Jyväskylä)
Avainsanat: Sähkökäytöt, integroitu järjestelmä, tarjousprosessi, käyttöjen mitoitus, tehontarvelaskelma, modulaarisuus, systeemin kehitys

Paperikoneen telojen linjakäyttöjen tehontarvelaskelma on osa paperikonetarjousta. Tehontarvelaskennan tarkoitus on määrittää laitteiden tehovaatimukset ja jakaa ne käyttöpisteille. Näiden tietojen avulla voidaan puolestaan valita kyseiselle paperikoneelle sopivat mekaaniset käyttölaitteet ja sähkömoottorit. Nykyisellä toimintatavalla yksi tarjousinsinööri ei pysty tekemään koko laskentaa itse, koska laskentaohjelmia on lukuisia ja ne noudattavat hyvin erilaisia toimintalogiikoita. Tämän työn tavoitteena on kehittää tehontarvelaskennan prosessi sellaiseksi, että se tukee nykyistä tehokkaammin linjalaajuista tarjoustoimintaa. Järjestelmästä pyritään muodostamaan sellainen, että se sitoo nykyistä vähemmän henkilöresursseja, sen ylläpito on hajautettu ja suunniteltava järjestelmä mukautuu ketterästi muuttuviin liiketoiminnan vaatimuksiin.

Työ jakautuu kahteen osaan: teoriaosaan ja soveltavaan osaan. Aluksi teoriaosassa tarkastellaan linjakäyttöjen tehontarpeen suhdetta paperikonelinjan energian kokonaiskulutukseen. Tämän jälkeen tutkitaan Metson tarjousprosessin nykyistä rakennetta, jotta prosessin vaatimukset osataan ottaa huomioon rakennettavan järjestelmän suunnittelussa. Seuraavaksi käydään läpi paperikoneen koneryhmien tyypilliset piirteet käyttöjen kannalta, jotta saavutetaan karkea ymmärrys laskennasta ja paperikoneen linjakäyttöjen tyypillisimmistä tehokaivoista. Lisäksi perehdytään karkeasti modulaarisuuden ja systeemien kehityksen teoriaan, minkä perusteella valitaan sopiva menetelmä, jolla järjestelmä kehitetään.

Työn soveltavassa osuudessa analysoidaan ensimmäiseksi rakenneryhmien nykyiset laskentapohjat ja käydään läpi käytettävyyden perusteita, mitkä antavat arvokasta tietoa järjestelmän arkkitehtuurin suunnitteluun. Tämän jälkeen esitetään suunnitellun järjestelmän arkkitehtuuri, rajapinnat, sekä käyttöliittymä. Järjestelmä todetaan alustavien testien perusteella toimivaksi, mutta järjestelmän intensiivisempää kehitystä jatketaan läpi laajemman käyttöönoton, jotta käyttäjien toiveisiin voidaan vastata nopeasti ja rakennetaan näin heidän luottamustaan järjestelmää kohtaan.

Lopuksi esitetään ehdotuksia järjestelmän edelleen kehittämiseksi: laskentapohjia olisi mahdollista yhdenmukaistaa nykyistä pidemmälle ja laskennan läpinäkyvyyttä voitaisiin lisätä käyttöluotettavuuden parantamiseksi. Toisaalta nyt kehitettyyn järjestelmään voitaisiin integroida myös muilla osastoilla käytettyjä laskentaohjelmia, jolloin vältyttäisiin päällekkäisyyksiltä ja vähennettäisiin kokonaisvaltaisesti ylläpidon työmäärää. Myös laskentapohjien kaavojen, toiminnan ja rakenteen dokumentointi kannattaisi yhdenmukaistaa nykyistä pidemmälle. Työn lopuksi esitetään ajatus lukuisia ohjelmia ja järjestelmiä palvelevasta ns. äitikonfiguraattorista, joka saattaisi mahdollistaa merkittäviä kokonaistehokkuusetuja, ja johon myös nyt kehitetty järjestelmä voitaisiin yhdistää.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Industrial Engineering and Management
IMMONEN, SAMPO: Paper machine wide integration of drive dimensioning
Master of Science Thesis, 92 pages, 0 Appendix pages
October 2012
Major: Product development
Examiner: Professor Asko Riitahuhta
Keywords: Drive power, integrated system, sales process, drive dimensioning, power requirement calculation, modularity

The power consumption calculation is part of the line wide quotation process. The purpose of the calculation is to determine the power requirements of the equipment and to distribute the required power to the drive points. This information will make it possible for the designers to determine which drives, gears and couplings are best suited for the paper machine. Currently there are a number of different calculation tools dedicated to each different section of the paper machine which the engineers can utilize to calculate the power requirements. As a result, today's calculation process is both challenging and slow because a single application engineer cannot possibly know all the calculation systems, and therefore cannot assemble a line wide power consumption requirement list by himself. The goal of this thesis is to find a better process for the above scenario, which will make the process faster, leaner, easier to upkeep and require fewer resources.

This thesis consists of two main parts: a theoretical section and a more practical, applied part. The theoretical part starts by comparing the drive power consumption to the line wide energy consumption. After this, a more in-depth inspection of Metso's current sales and quotation process is presented. Metso's process reveals some of the necessary requirements for the system designing. Then the thesis will discuss the various yet typical features of the paper machine sections respective to the power consumption calculation. These features provide an understanding to the power calculation and the most common power sinks. The final section discusses the basics of modularity and system engineering, which determinates applicable development process for the system.

The applied section of this thesis starts by analyzing the current calculation tools and going through the basics of system usability, which gives valuable information for designing the system architecture. The next few chapters of the thesis discusses the system architecture, the internal interface and the user interface from the developed system. According to the preliminary tests, the system is functional, however even though the system is past its most intensive points of development; it is by no means at its end. The development will continue throughout the wider launch of the program. This creates the possibility to react fast to the development ideas and wishes of the users during and after the launch. Creating such an user-friendly environment, that takes into account the user's feedback, will build up a certain confidence towards the system.

The final part of this thesis will present a few ideas for further development of the system: calculation tools could be made more similar and the transparency of the calculations could be improved to achieve a better user experience. Another possibility is to integrate several other calculation tools into the developed system from other departments, which would reduce the overlapping upkeep. Also calculation tool formulas, logic and documentation could be made more similar compared to the current situation. Lastly an idea concerning a "mother-configurator", which would not only serve many different calculation tools, but also the system discussed throughout this thesis.

SISÄLLYS

TERMIT JA NIIDEN MÄÄRITELMÄT

API	Application Programming Interface = ohjelmallinen rajapinta
HCI	Human Computer Interface = Käyttöliittymä
Hit-ratio	Myynnissä käytettävä mittari, jolla vertaillaan kuinka paljon kauppoja saadaan suhteessa tehtyjen tarjousten määrään
I/O	Input/output = sisääntulo/ulostulo
IBB	Installed Base Business = Pienempää paperikoneen tehostamiseen tähtäävää toimintaa
Käyttölaitteet	Laitteita, jotka kuuluvat mekaanisen käytön voimansiirtoketjuun (kts. Mekaaninen käyttö).
LWC	Light weight coated = kevyt, päällystetty paperi.
Mekaaninen käyttö	Tarkoittaa laitteiden ja komponenttien voimansiirtoketjua, jolla moottorin teho välitetään käytettävälle laitteelle (= telalle).
Nippi	Nipillä tarkoitetaan puristavaa aluetta, johon raina joutuu, kun se ajetaan kahden telan välistä.
NRL	Normal Running Load, normaalin ajotilanteen käyttötehontarve.
Raina	Rainalla tarkoitetaan paperintekoprosessissa kulkevaa paperirataa.
RDC	Recommended Drive Capasity, tilapäisten kuormitusolosuhteiden maksimitehontarve.
SC	Super calendered = moninippikalanteroitu.
Sähkökäyttö	Tarkoittaa sähkömoottoria ja sen ohjauksen yhdistelmää.

SDLC	System Development Life Cycle = systeemin kehityksen elinkaari.
UD	Usability designer = Käytettävyydensuunnittelija
UI	User Interface, käyttöliittymä käyttäjän näkökulmasta.
UML	Unified Modeling Language = Järjestelmä- ja systeemikehitystä varten alunperin kehitetty graafinen mallinnuskieli
WFC	Wood free coated = kemiallisesta sellusta tehty päällystetty paperi
WFU	Wood free uncoated = kemiallisesta sellusta tehty päällystämätön paperi

1. JOHDANTO

Luotettava ja nopea tarjoustyö on tärkeä osa myyntitoimintaa. Metso Paperilla tarjouksia tehdään vuodessa satoja, joten tarjouksesta voidaan projektitoimintaan keskittyneessä organisaatiossa puhua volyymituotteena. Tarjoukseen kuuluu lukuisia erilaisia dokumentteja ja selvityksiä, joiden tekemiseen kuluu aikaa. Mikäli näiden dokumenttien tekemistä ja selvitystä voidaan nopeuttaa, voidaan asiakasta palvella nopeammin ja vähemmän omaa organisaatiota rasittaen. Paperikoneen tehojen määrittely on yksi osa tarjousprosessia ja tämän tehomäärittelyprosessin kehittäminen on tämän työn päätarkoitus.

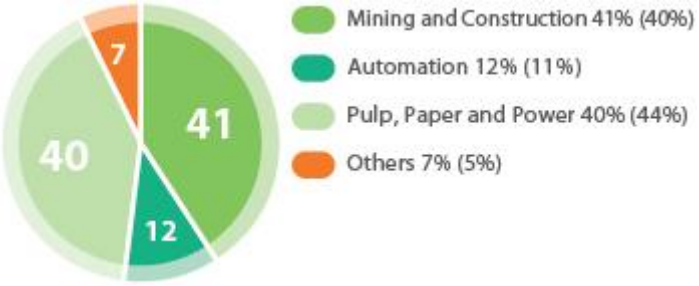
Nykyisellään toiminnassa on monenlaisia haasteita. Tällä hetkellä ylläpito ja käyttö on kenties hallittua, muttei kokonaisuuden kannalta. Tavoitteena ei kuitenkaan ole täysin keskitetty ylläpito, vaan hallitun hajautetun ja keskitetyn ylläpidon välimuoto – ns. pseudo-hajautettu ylläpito. Tarkemmin eriteltynä tällä tarkoitetaan sitä, että ohjelmiston kokonaiskäytettävyys ja logiikka pysyy keskitetyssä ylläpidossa, mutta yksittäisten laskentapohjien ylläpito olisi hajautettu laskentaryhmiin. Lisähaasteita tuo myös nykyiseltään haastava laajennettavuus, sekä päivitettävyys ja toisaalta myös jatkuvasti lisääntyvät mitoituskohteet uusien tuotteiden ja komponenttien myötä.

Työn tavoitteena on selvittää Metson linjakäyttöjen mitoitusprosessia koko linjan tasolla ja selvityksen pohjalta kehittää ohjelmistotyökalu, jonka avulla myyntiprojektin kaikki linjakäytöt voidaan mitoittaa yhdestä paikasta ja yhden henkilön toimesta luotettavasti ja nopeasti. Tämän lisäksi sivutavoitteena on tutkia ja mahdollisuuksien mukaan yhdenmukaistaa nykyisten, hyvin erilaisten laskentapohjien logiikkaa ja laskuperusteita. Kehitettävältä systeemiltä toivotaan helppoa käytettävyyttä ja ylläpidon helppoutta, koska ylläpito on tarkoitus jalkauttaa suunnitteluorganisaation puolelle, jossa viimeisin laskentatieto sijaitsee. Työn ohessa yritetään pitää mielessä myös sellukoneiden kehitystarpeet käyttöjen näkökulmasta, koska sellutoimintaa on osittain siirretty saman katon alle. Tämän lisäksi toiveena on myös, että tehtävä selvitys ja työkalu selkiyttäisivät myös tiedonsiirtoa myynnistä toimitusprojektille, koska tällä hetkellä vastuun jakautuminen näiden vaiheiden rajapinnassa asettaa haasteita.

1.1. Metso lyhyesti

Metso on kansainvälinen teknologiakonserni, jonka erikoisosaamista ovat teknologia- ja palveluratkaisut. Liiketoiminta on organisoitu kolmeen segmenttiin:

- kaivos- ja maarakennus
- automaatio
- massan-, paperin- ja voimantuotanto



***Kuva 1.1** – Liikevaihdon jakautuminen Metson liiketoimintayksiköiden välillä*

Metso on aidosti globaali yritys – sillä on suunnittelua, tuotantoa, hankintaa, palveluliiketoimintaa, myyntiä ja muuta toimintaa yli 300 yksikössä yli 50 maassa. Metso työllistää maailmanlaajuisesti noin 30 000 ihmistä ja asiakkaita on yli 100 maassa.

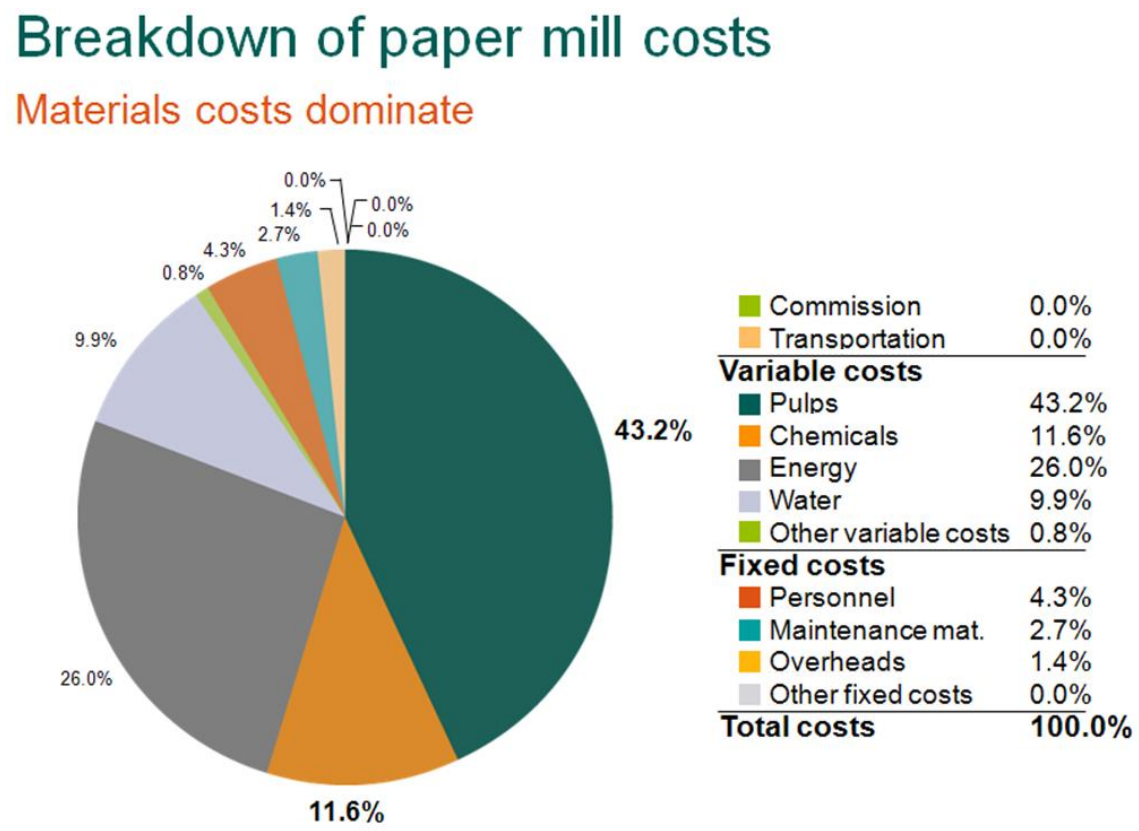
Vuonna 2011 Metso-konsernin liikevaihto oli 6 646 miljoonaa euroa ja 45 prosenttia liikevaihdosta muodostui palveluliiketoiminnasta. [Metso 2011]

Huomattavaa Metson paperikonetoiminnan kannalta on sen projektiluonteisuus. Isoja, koko linjan kattavia toimituksia on tyypillisesti noin tusina vuodessa. Pienempiä uusintaprojekteja, voi olla joitain kymmeniä. Pieniä komponenttitoimituksia sen sijaan voi olla satoja. Tämä toimituslaajuuksien ja frekvenssien moninaisuus heijastuu myös Metson systeemeihin, joiden pitää soveltua toisaalta isoihin, puhtaisiin projektitoimituksiin kuin myös melko rutiininomaisiin pienempiin toimituksiin.

1.2. Paperikoneen energian kulutusarvolaskenta

Tänä päivänä kulutusarvot ja tarkemmin sanottuna energiatehokkuus on yhä tärkeämmässä roolissa yritysten vihertyvissä arvoissa myös perinteisemmässä perusteollisuudessa - kuten paperiteollisuudessa. Tästä hyvänä esimerkkinä esimerkiksi StoraEnson uudet teemasanat (rethink, redefine, redesign, recycle, rebuild, revive ja reimagine [StoraEnso 2011]), jotka kaikki tavalla tai toisella liittyvät kestävään kehitykseen. Metsokin on aina ollut asiakaslähtöinen yritys, joten tämä asiakaslähtöinen ympäristöpaine kohdistuu myös Metsoon itseensä, jotta Metso voi toimittaa paperinvalmistuslinjoja, jotka täyttävät yhä tiukentuvia energiatehokkuuden vaatimuksia. Paperinvalmistajien kannalta energiatehokkuuden korostaminen on kaiken kaikkiaan ”win-win”-tilanne, koska sen avulla ne voivat kiillottaa julkisuuskuvaansa ja toisaalta laskea operatiivisia käyttökustannuksia, jotka vastaavat pääosasta paperikoneen elinkaarikustannuksista (laitteiston investointikulut ovat pitkässä juoksussa murto-osa (~10%) elinkaarikustannuksia). *Ku-*

vasta 1.2 nähdään, että kokonaisuudessaan energian kulutus vastaa yli neljännestä koko paperikoneen kustannuksista. Energia on toisiksi suurin yksittäinen kuluerä heti sellun jälkeen.



***Kuva 1.2** – Kustannusten jakautuminen paperitehtaassa [Ketolainen 2011]*

Jotta voidaan ymmärtää linjakäyttöjen kulutuksen asema osana isompaa kuvaa, on hyvä ymmärtää mistä paperikoneen energiankulutus kokonaisuudessaan muodostuu. Paperikoneen energiankulutus voidaan jakaa kolmeen kategoriaan: sähkön, höyryn ja polttokaasun kulutukseen.

Paperikoneella energiaa kuluu ennen paperikonetta tasalaatuisen ja riittävän puhtaan massan valmistukseen, sekä pumppaukseen. Paperikoneella suuri osa energiasta kuluu hallittuun vedenpoistoon paperiradasta. Vedenpoiston volyymistä saa osviittaa seuraavasta esimerkkilaskusta. Kun radan kuiva-ainepitoisuus (KAP) muuttuu perälaatikolta 0,5 prosentista loppurullan 97% KAP:iin, tarvitsee radasta poistaa noin 200 tonnia vettä jokaista tuotettua paperitonnia kohden. Jälkikäsittelyssä energiaa tarvitaan tuotteen ominaisuuksien ja laadun parantamiseen.

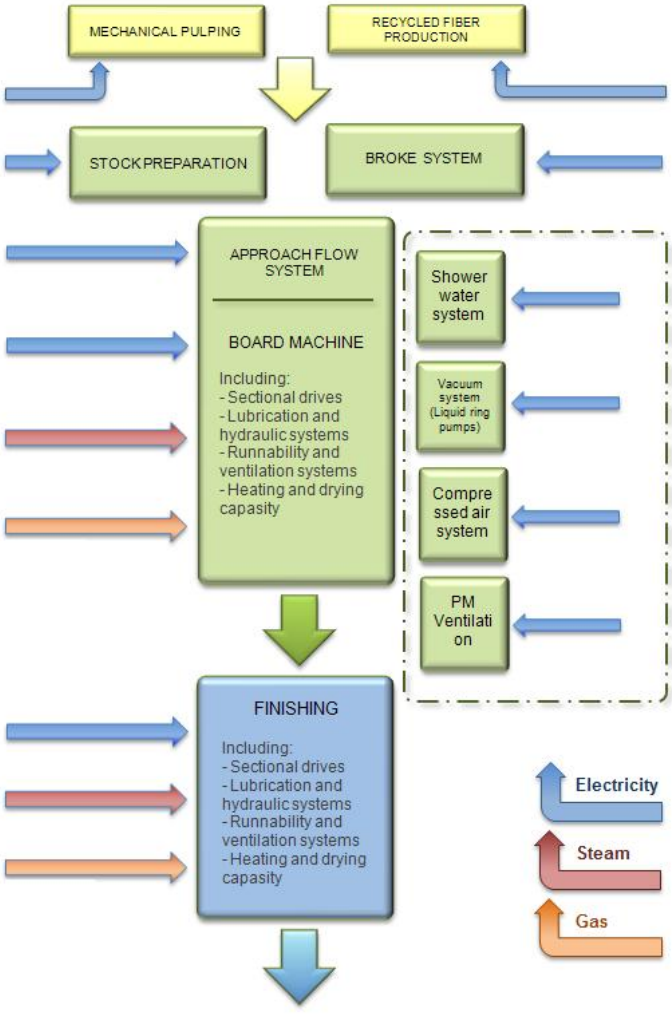
1.2.1. Sähkönkulutus

Paperin valmistusprosessissa sähköä kuluu kaikissa vaiheissa valmistusprosessia. Suurimpia sähköenergian kuluttajia ovat mekaanisen kuidunvalmistuksen laitteet, joilla tukeista tehdään hioketta tai hierrettä. Myös fluidimaisten massojen pumppaus vaatii huomattavan suuria sähkötehoja. Näiden lisäksi suuri määrä sähköenergiaa kuluu pape-

rikoneen linjakäyttöihin, eli käytännössä telojen pyöritykseen, joihin tämä nimenomainen diplomityö keskittyy.

Sähköenergiaa tarvitaan [Ketolainen 2002]:

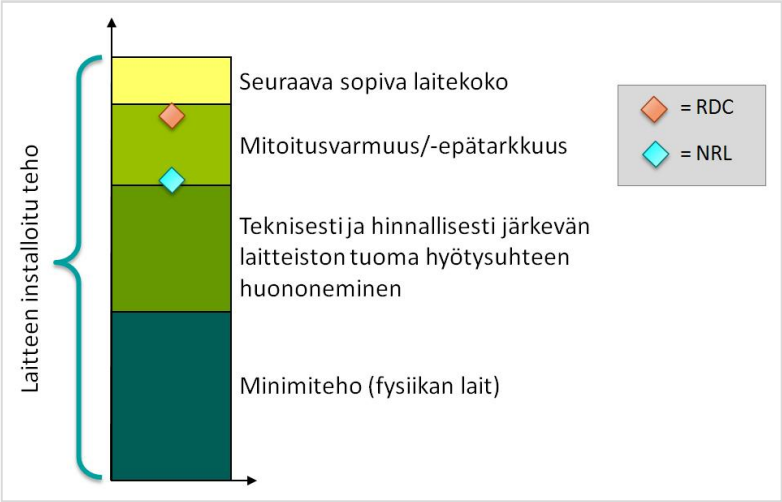
- Telojen ja sylinterien pyörittämiseen
- Massan ja veden pumppaamiseen
- Veden poistoon (tyhjö, infrat, puhaltimet, ilmastointi)
- Kudoksia laahaavien komponenttien ja kaavareiden aiheuttaman kitkan voittamiseen
- Puristusnipeissä muodonmuutostyöhön
- Paperin vetämiseen ja ratakireyden ylläpitoon



***Kuva 1.3** – Paperinvalmistusprosessissa energian käyttö eri puolilla paperikonetta*

Kuvassa 1.3 on kaaviomuotoisesti esitetty karkealla tasolla kuinka eri energiamuotoja käytetään eripuolilla paperikonetta. Kuvassa sinisillä nuolilla on merkitty sähköä kuluttavat kohteet, punaisella höyryä kuluttavat kohteet ja oranssilla kaasua kuluttavat kohteet. Paperi-/kartonkikoneen mekaaniset käytöt sisältyvät kuvassa ”board machine”:een ja ”finishing”:iin meneviin sinisiin nuoliin. Isoja käyttöjä löytyy myös

massan valmistuksesta, mutta ne on jätetty tämän työn rajauksen ulkopuolelle, koska niitä ei lasketa linjakäyttöihin.

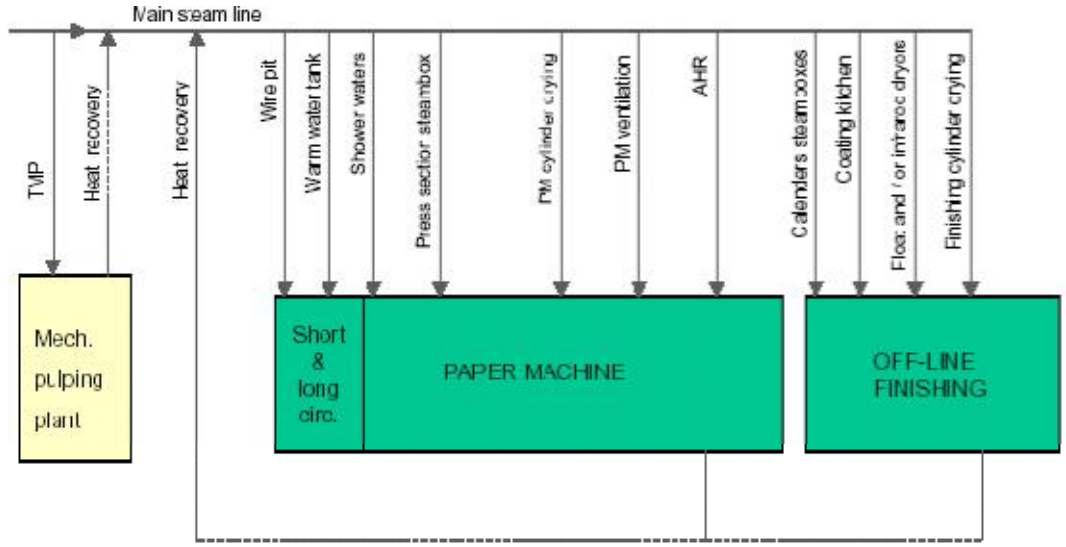


***Kuva 1.4** – Mitoitusvarmuuden skemaattinen esitys*

Kuvassa 1.4 on esitetty skemaattisesti kuinka tehollinen kokonaismitoitus muodostuu. Perusteena mitoitukselle on fysiikan lakien määräämä minimitehovaatimus. Koska mikään systeemi ei kykene täydelliseen hyötysuhteeseen, eikä se olisi teknistaloudellisesti kannattavaa, tulee minimitehon päälle hyötysuhteen vaatima ”ylimitoitus”. Tämän lisäksi hyvien insinöörikäytäntöjen mukaisesti lisätään kokonaistehoon vielä mitoitusvarmuus, jolla varmistetaan, että laite täyttää vaatimukset. Näiden kolmen komponentin määräämän tehon perusteella etsitään seuraava sopiva laitekoko, joka täyttää tehovaatimukset. Näiden elementtien muodostamaa lukemaa kutsutaan laitteen installoiduksi tehoksi. Kuvaajasta voi nähdä kuinka normaaliajotilanteen (NRL) ja erityisemmän ajotilanteen (RDC) vaatimat tehot sijoittuvat kokonaismitoitukseen – tyypillisesti mitoitusvarmuuden sisälle.

1.2.2. Höyrynkulutus

Suurin osa prosessin kuluttamasta höyrystä käytetään paperiradan lämmitykseen ja kuivatukseen. *Kuvassa 1.5* on esitetty prosessin höyryn merkittävimmät käyttökohteet. Höyryä kuluu energiamielessä paperikoneella huomattavia määriä – energiaekvivalenttina laskettuna jopa yli kolme kertaa sähköä enemmän (kts. *kuva 3.22*). Prosessilaitteet on kuitenkin pyritty rakentamaan niin, että suurin osa tuodusta lämpöenergiasta saadaan uudelleenhyödynnettyä muun muassa prosessivesien ja tehdastilojen lämmitykseen. [Ketolainen 2002]



***Kuva 1.5** – Höyryn käyttökohteet paperikoneessa [Ketolainen 2002]*

Merkittävin höyrynkulutukseen vaikuttava tekijä on rainasta poistettava vesimäärä. Nyrkkisääntönä voidaan sanoa, että jokainen prosentti lisää kuiva-ainetta, joka saavutetaan puristimen loppuun mennessä säästää 4 % höyrynkulutuksessa kuivatusosalla. Vaikka höyryn käyttö osana kuivatusta vaatii paperikoneelta huomattavasti enemmän tilaa kuin kaasu- tai sähkökuivaavat vastineet, niin höyryn hyötysuhde (~80-85% vs sähkön 40%) ja hinta (kolmasosa sähkön hinnasta) antavat höyrylle huomattavan kilpailuedun, joka yleensä ylittää tilankäytöllisen haitan. Kuivatuksen lisäksi höyryä käytetään prosessissa esimerkiksi paperiradan poikkisuuntaisen kosteusprofiilin säätöön.

1.2.3. Polttokaasun kulutus

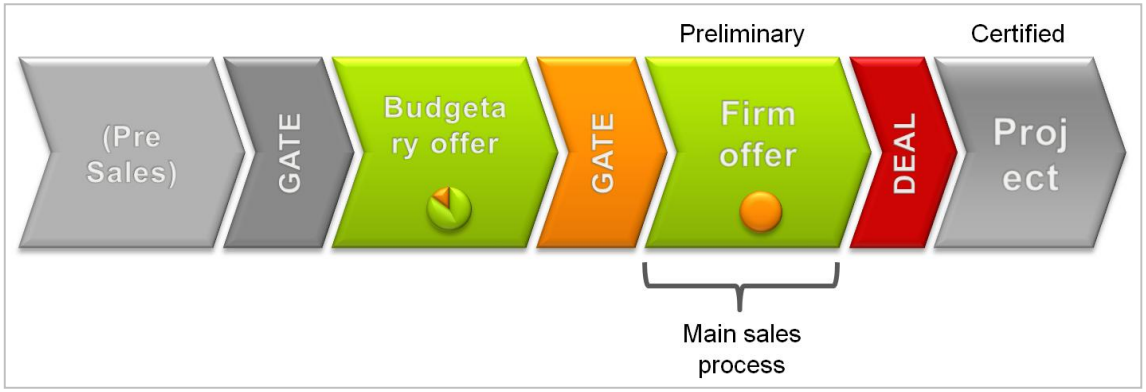
Paperikoneen polttokaasuna voidaan käyttää maakaasua ja nestekaasua useissa eri muodoissa, riippuen siitä mitä on alueellisesti parhaiten hyödynnettävissä ja edullisimmin saatavissa. Polttokaasuja käytetään tyypillisesti kosketuksettomissa kuivaimissa, kuten infra-kuivaimissa, päällepuhallusyksiköissä, leijuissa, sekä kaasuleijuissa. Näiden toimintalogiikka perustuu siihen, että kaasun polttaminen vapauttaa lämpöä, joka lämmittää ilmaa, joka puolestaan ohjataan puhalluksella paperirataan. Tyypillisin paikka näille kuivaimille on paperikoneen päällystyksen tai liimauksen jälkeiset positiot, joissa haihdutettava vesimäärä ei ole yhtä suuri kuin etukuivaimella ja toisaalta kuivauksen tulee olla hyvän ajettavuuden takia kosketukseton. [Ketolainen 2002]

2. TARJOUSPROSESSI

Myynnin tarjousprosessi on iso kokonaisuus, joka elää läpi projektin. Tässä kappaleessa selvitetään tarkemmin kuinka käyttöjen mitoitusprosessi nivoutuu myynnin tarjousprosessiin ja projektin muihin vaiheisiin.

2.1. Metson myyntiprosessi

Jotta voidaan ymmärtää mekaanisten käyttöjen mitoituksen prosessia, on hyvä ymmärtää myös kuinka se suhteutuu myynnin kokonaisprosessiin. Tämän avulla voidaan päätellä vaatimuksia, joita työkalun pitää täyttää ja ominaisuuksia, jotka ovat työkalun käytön kannalta hyödyllisiä.



***Kuva 2.1** – Metson myyntiprosessin flow chart [Kinnunen 2010]*

Kuvassa 2.1 on nähtävissä skemaattisesti Metson nykyinen myynnin kokonaisprosessi. *Pre sales* tarkoittaa vaihetta, jossa markkinoita ja asiakkaiden tarpeita tarkastellaan pro-aktiivisesti sopivien tarjouskohteiden löytämiseksi ja valitsemiseksi. Tällöin voidaan esimerkiksi tehdä business-analyysejä, joiden avulla voidaan selvittää asiakkaiden, sekä alueiden investointipotentiaalia. Tämän perusteella markkinointia ja lähempää asiakastyöskentelyä voidaan suunnata mahdollisimman tehokkaasti. Pre sales sisältää myös vanhojen projektien seuraamista, jotta voidaan kartoittaa linjojen päivitystarpeita ja kaupata pienempiä IBB-projekteja. Pre sales-vaiheessa on suuri määrä asiakkaita ja potentiaalisia projekteja. Jotta projekti voi siirtyä pre sales vaiheesta budjettitarjousvaiheeseen, täytyy sen läpäistä portti, jonka ehtona on se, että myyntiprojekti täyttää vaaditut vaatimukset.

Budjettitarjousvaihessa (*budgetary offer*) asiakkaita autetaan investointiprojektin suunnittelussa ja budjetin laatimisessa Pre sales-vaihetta tarkemmin. Tällöin kartoite-

taan yhdessä asiakkaan kanssa potentiaaliset konseptit ja vaihtoehdot, joiden pohjalta projektia kannattaa lähteä viemään eteenpäin. Tässä vaiheessa tehdään suuntaa antavia kulutuslaskelmia ja toimituslaajuuden kuvaus päätasolla, sekä tämän hinnoittelu, jotta investointilaskelmia voidaan tarkentaa. Tästä vaiheesta eteenpäin siirtymiseksi projektin pitää täyttää etukäteen asetetut tavoitteet.

Sitovan tarjouksen (*firm offer*) vaiheessa konekonsepti on valittu asiakkaan kanssa, hinnoittelu on tarkentunut ja tekniset erittelyt on tehty. Tässä vaiheessa mukana on usein myös kilpailijat omien tarjoustensa kanssa. Tämä vaihe päättyy käytännössä siihen, että joku toimittaja saa kaupan (*deal*). Jos Metso saa kaupan, projekti siirtyy toimitusprojektivaiheeseen ja projektipäällikön käsiin. Tästä eteenpäin myynnin vastuu projektista on lähinnä tukemisen muodossa. [Kinnunen 2010]

2.1.1. Tarjouksen laadinta

Tarjouksen laadinta paperikonelinjan kokoisessa projektissa on työlästä. Tarjous on itsessään lainvoimainen dokumentti ja siksi sen tekeminen ei ole triviaalia. Tarjouksen tekemiseen osallistuu tyypillisesti projektista vastaava myyntipäällikkö, tarjousinsinööri, teknisiä assistentteja, sekä joukko muilta osastoilta nimettyjä myynnin tukihenkilöitä. Kaiken kaikkiaan tarjouksen tekoon voi osallistua yhteensä jopa yli kymmenen eri sisäistä yksikköä tai toimintoa, jotka kaikki antavat osan tarjouksessa vaadittavista tiedoista. Näiden tietojen pohjalta myyntiosasto rakentaa asiakkaalle yhden yhtenäisen tarjouksen. Tarjous itsessään voi sisältää jopa 1500 sivua dokumentteja, jotka sisältävät erilaisia kuvauksia prosesseista, laitteista ja järjestelmistä. [Ketolainen 2002] Niissä esitetään mitoitukseen liittyviä tietoja, prosessikaavioita, laitelistoja, vastuujakoja yms. Näiden dokumenttien seassa on myös linjakäyttöjen tehontarveluettelo, joka on asiakkaan suuntaan linjakäyttöjen näkökulmasta rajapintadokumentti.

Esimerkin vuoksi mainittakoon, että vuonna 2011 Metson myyntiosasto lähetti asiakkaille 542 tarjousta, joista sitovia tarjouksia oli 228. Sitovista tarjouksista 51 kpl koskivat uusia paperikonelinjoja, loput olemassa olevien linjojen modernisointeja tai yksittäisten rakenneryhmien korvaamista uusilla [Poikonen 2012]. Tänä päivänä painopistettä tarjousten suhteen on pyritty siirtämään mahdollisimman paljon sitovasta tarjousvaiheesta myyntiprosessin alkupäähän. Tällä pyritään välttämään ”turhaan” tehtyä tarjoustyötä, joka puolestaan parantaa hit-ratiota, jota voidaan pitää yhtenä tehokkaan myynnin mittarina. Tämän ansiosta myynnissä voidaan keskittyä paremmin lähitulevaisuudessa päätökseen meneviin projekteihin. Tarjoukset jotka hävitään syövät merkittävän määrän työtä, eikä niistä saada rahaa, joka syö yrityksen katetta ja heikentää kilpailukykyä. Tämän takia prosessin sulavoittaminen entisestään on tärkeää. Siihen tämäkin työ osaltaan tähtää.

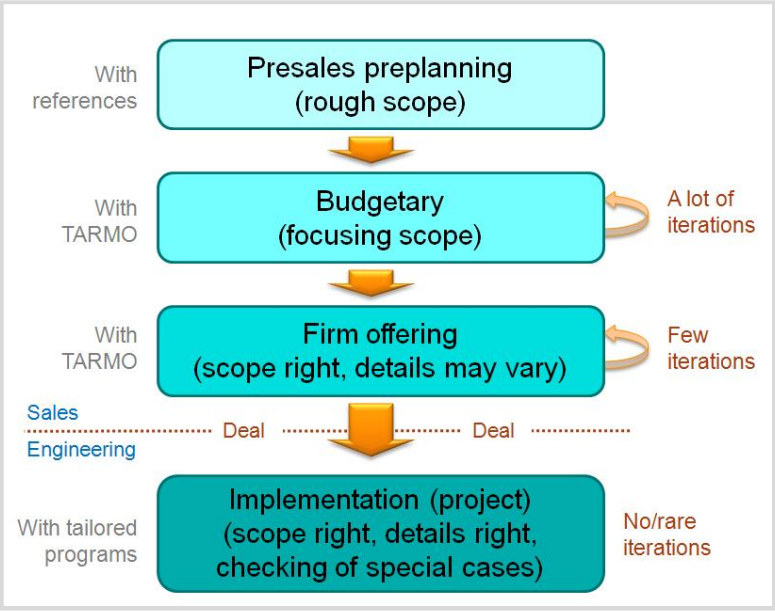
2.2. Nykyinen prosessi linjakäyttöjen näkökulmasta

Kuvasta 2.2 nähdään kuinka linjakäyttöjen mitoitus tällä hetkellä suhteutuu myynnin prosessiin. *Pre sales* vaiheessa mitoitustietona käytetään pääsääntöisesti referenssipro-

jekteja, joiden perusteella päästään kunkin konseptin osalta hyvin oikealle dekadille käyttötehojen suhteen, eikä linjakäyttöjä tarvitse tässä vaiheessa käytännössä laskea.

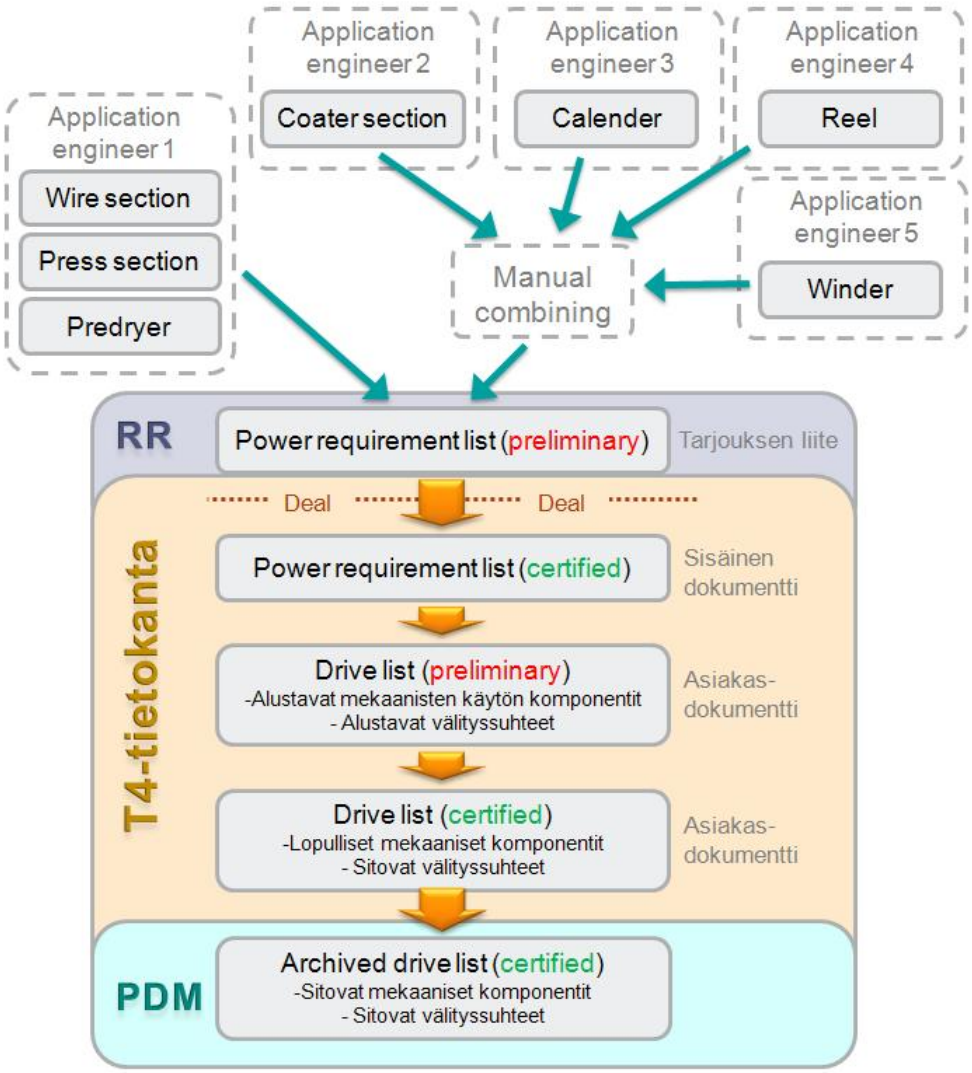
Budjetti-vaiheen projekteissa käytetään jo tarkempia mitoituslaskelmia. Yksinkertaisuuden vuoksi niitä kutsutaan kuvassa yhteisnimikkeellä ”*With TARMO*”, vaikka tällä hetkellä vain viira-, puristin- ja kuivatusosa lasketaan Tarmo-ohjelmalla. Tässä vaiheessa suuri määrä teholistan iteraatioita on tyypillistä, kun asiakkaan kanssa etsitään aktiivisesti parasta teknistä ratkaisua asiakkaan tarpeisiin. Tällöin yksittäiset laitteet ja jopa koko ratkaisu ”elää”, jonka seurauksena myös tehontarpeet joudutaan päivittämään kutakin ratkaisua vastaavaksi, jotta näitä voidaan vertailla. Suuren toistomäärän takia on tärkeää, että laskenta olisi mahdollisimman nopea käyttää ja muokata. Kokonaisprosessin näkökulmasta tämä ei tällä hetkellä toteudu – koko laskelman päivittäminen on työlästä, koska laskenta tehdään hajautetusti jokaiselle rakenneryhmälle.

Kiinteän tarjouksen vaiheessa jatketaan yhä Tarmon (ja hajautetun laskennan) avulla. Tällöin iteraatioita ei ole enää niin paljoa, koska ratkaisu ja laajuus on tarkentunut lopulliseen tarkkuuteensa.



***Kuva 2.2** – Teholistan iterointi myyntiprojektin eri vaiheissa, sekä toimitusprojektissa*

Jos asiakas hyväksyy kiinteän tarjouksen ja kauppa tehdään, siirtyy myyntiprojekti *toimitusprojekti*-vaiheeseen. Tässä vaiheessa tehontarpeet toimitetaan toteutusprojektiorganisaation ja suunnittelun vastuulle, jossa ne tarkastetaan. Mikäli korjattavaa löytyy, korjaukset tehdään tyypillisesti suunnitteluosastojen omilla räätälöidyillä ohjelmilla, jotka sisältävät yleensä viimeisimmän laskentatiedon. Tämä on kuitenkin työläs prosessi, koska lähtötiedot joudutaan etsimään ja syöttämään uudelleen, jonka takia olisi tärkeää, että tehotiedot olisivat jo lähtökohtaisesti toteutuskelpoiset.



Kuva 2.3 *– Nykyisen tehontarveluettelon ja käyttölaitteiden listan elinkaari*

Kun tehontarveluettelon ja käyttöluettelon elinkaarta tarkastellaan laajemmin, niin saadaan parempi kuva siitä, että miten dokumentti syntyy, kuinka se kehittyy ja kuinka se ”kuolee”. *Kuvassa 2.3* on nähtävillä kaavio, josta elinkaari ja kulloisetkin dokumentin revisiointi ja arkistointimenetelmät löytyvät. Projektin ensimmäinen alustava tehontarveluettelo muodostetaan kuvan mukaisesti viiden eri tahon yhteistoimesta. Jyväskylän tarjousinsinööri vastaa viira-, puristin- ja kuivaosuuksien tekemisestä nykyisen Tarmon avulla. Päällystyksen, kalanterin, rullaimen ja leikkurin osalta listat tekee neljä eri henkilöä, jotka ovat kyseisten alueiden tarjousinsinöörit. Näistä viidestä dokumenteista kasataan manuaalisesti kokonaistehontarveluettelon, joka laitetaan tarjouksen liitteeksi, sekä myynnin tietokantaan (Road Runner = RR). RR on myyntivaiheessa käytettävä projektidokumenttien säilytykseen ja jakamiseen tarkoitettu tietokanta. *Kuvan 2.3* selkeyttämiseksi kaaviosta on jätetty pois *kuvassa 2.2* esitetyt iteraatiokierrokset. Mikäli iteraatioita ennen kauppaa tulee, ne tekee kukin laskentataho itse, jonka jälkeen kokonaislista kasataan uudelleen päivitettyjen laskelmien pohjalta.

Kun alustava tehontarveluettelo on hyväksytty ja mahdollinen kauppa saatu, seuraavassa vaiheessa on vuorossa tehontarveluettelon vahvistus (*certified*). Tässä vaiheessa lista siirretään Tasman-projektikantaan (*T4*), joka on toteutusprojektissa käytettävä tietokanta, joka toimii suunnitteluvaiheen dokumenttien sijoituspaikkana. Tämä on kanta, jossa listaa päivitetään sisäisesti, kunnes se todetaan lopulliseksi. Lopullisen tehontarveluettelon perusteella kootaan käyttöluettelo (*Drive List*), joka sisältää käyttöpisteiden tehotietojen lisäksi myös käyttöpistekohtaiset mekaanisten käyttöjen tiedot, eli:

- Välitystiedot
- Moottoritiedot
- Mekaanisen käytön komponentit
- Perustustarvikkeet

Käyttöluettelo on asiakasdokumentti, jota tarvittaessa iteroidaan asiakkaan kanssa yhteistyössä. Kun alustava drive list on hyväksytty lopulliseksi, niin dokumentti luovutetaan asiakkaalle ja/tai hankinnalle, jonka perusteella tarvittavat käyttölaitehankinnat tehdään.

Asiakkaalle luovutuksen jälkeen käyttöluettelon viimeinen versio arkistoidaan Metson PDM-järjestelmään.

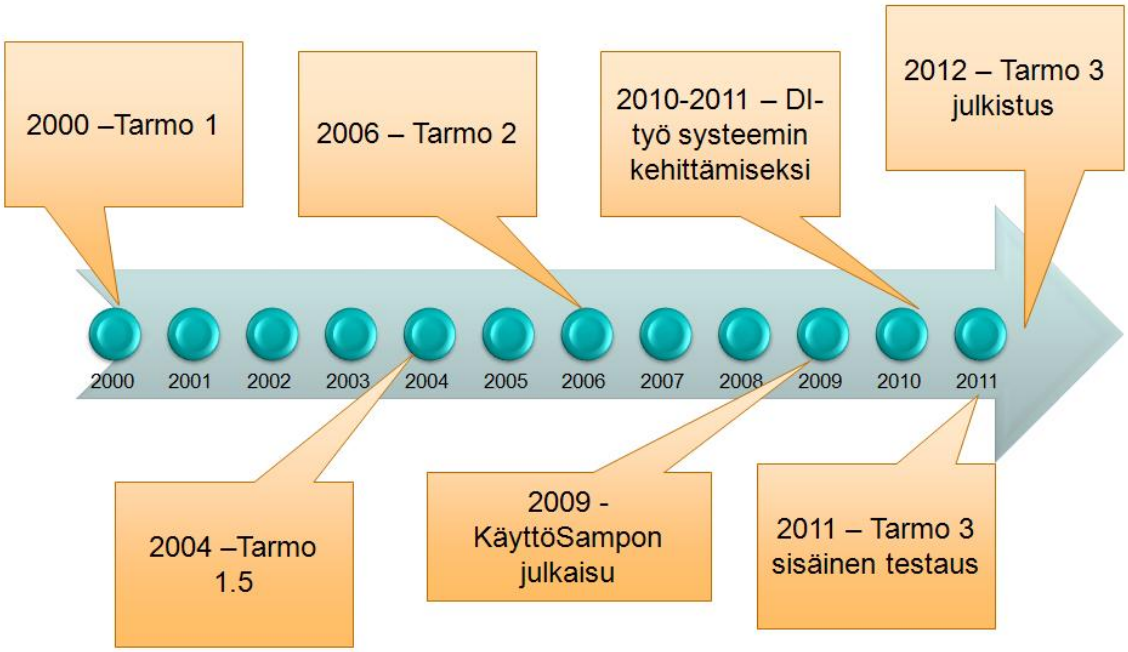
2.3. Aiempi tehontarvelaskenta - Tarmo 2

Nykyinen Metso Paper Jyväskylässä käytettävä linjakäyttöjen tehontarvelaskentaa on kehitetty jo pitkään ennen kuin se on saavuttanut nykyisen muotonsa (*kuva 2.4*). 2000-luvun alussa tehtiin ensimmäinen Tarmon versio, joka keräsi yhteen hajanaista laskentatietoa silloisista konsepteista. Ulkoisesti ohjelma oli toteutettu MS Excelin formeilla. Käyttö oli sinänsä helppoa, koska muutettavia asioita oli melko vähän. Muutoin käyttö ei ollut nykystandardeilla intuitiivista tai käyttöliittymä silmää miellyttävä. Ohjelma oli kuitenkin huomattava harppaus suhteessa vallinneeseen tilanteeseen.

Vuoteen 2004 mennessä Tarmo oli edennyt versioonsa 1,5. Tässä versiossa käyttöliittymä oli säilytetty yhä samassa kuosissaan, mutta laskentaan oli lisätty enemmän mahdollisuuksia mitoitettavien rakenneryhmien variointiin. Vasta vuonna 2006 Tarmo koki ensimmäisen suuremman uudistuksensa. Tällöin ohjelma rakennettiin kokonaan uudelleen erilaisen käyttöliittymän päälle. Uusi käyttöliittymä suunniteltiin myyntiosaston toimesta teetetyn ”Investointi-laskentasovelluksen käyttäjäkeskeinen suunnitteluprosessi” [Palomäki & Puurula 2005] Pro gradu-tutkielman tuloksia myötäillen. Tätä käsitellään tarkemmin kappaleissa ”5. Teoria - Systeemien käytettävyys” ja ”7.2 Ajatuksia käyttöliittymästä”. Tämän lisäksi uusi versio esitteli myös entistä laajemman linjan varioinnin mahdollisuuden. Uuden version helpomman käytettävyyden ja paremman varioitavuuden ansiosta ohjelma ansaitsi käyttäjäkunnan luottamuksen.

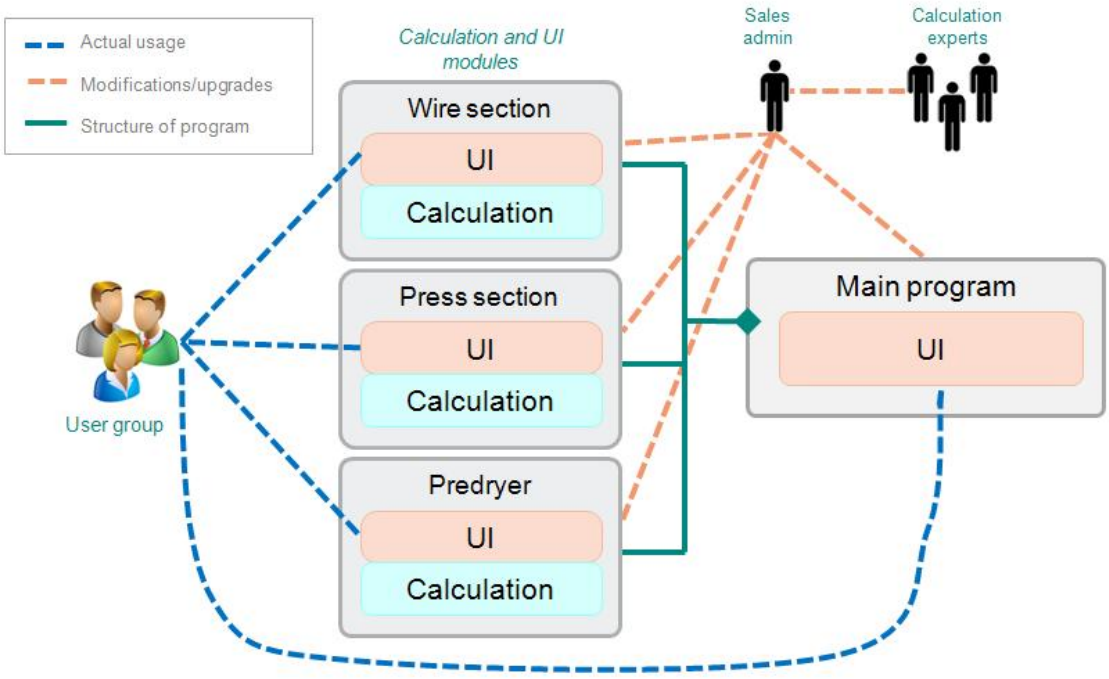
Vuonna 2009 julkaistiin Tarmon tuloksia suoraan hyödyntävä KäyttöSampo, joka muodostaa tehontarveluettelon perusteella automaattisesti käyttöpistekohtaisen käyttöluettelon. Toisin sanoen ohjelma mitoittaa ja valitsee välityssuhteet, vaihteet, kytkimet

ja käyttöjen perustustarvikkeet. Näiden avulla saadaan laskettua budjettitason kustannukset kyseisille komponenteille. Ohjelmaa tehdessä suurimmaksi haasteeksi osoittautuivat linjatasolla erilaiset teholistat. Taustasyyksi selvisi se, että tehontarvelaskentaa ei laskettu vielä linjatasolla yhtenäisesti. Tämä oli yksi tämän diplomityön syntymisen katalysaattoreita.



Kuva 2.4 *– Tarmon historia*

Kuvassa 2.5 on esitetty Tarmo2:n karkea rakenne ja kuinka rakenne suhtautuu eri käyttäjäryhmiin. Tarmo2 rakentuu karkeasti siten, että se sisältää pääohjelman, sekä puoli-itsenäiset laskentamoduulit. Puoli-itsenäisellä tarkoitetaan tässä tapauksessa sitä, että laskentamoduuli sisältää laskennan lisäksi myös siihen liittyvän käyttöliittymän, käyttölogiikan, sekä oletusarvot. Pääohjelma toimii ikään kuin pääohjelmana, jolla kutsutaan ja ohjataan puoli-itsenäisiä laskentoja. Pääohjelma itsessään sisältää vähemmän toiminnallisuutta, eikä se puutu laskentaan laisinkaan.



***Kuva 2.5** – Tarmo2:n toimintarakenne*

Käyttäjäryhmät linkittyvät kuvan osoittamalla tavalla kokonaisuuteen. Ohjelman käyttäjät – eli pääasiassa tarjousinsinöörit – käyttävät ainoastaan ohjelman käyttäjärajapintoja, eivätkä näe taustalla olevaa laskentaa ja toiminnallisuutta. Myynnin ylläpitohenkilö puolestaan on linkittynyt pääohjelmaan ja kaikkiin moduuleihin. Hän on vastuussa ohjelman kokonaisvaltaisesta ylläpidosta ja hänen kauttaan toteutetaan niin käyttöliittymän, toiminnan kuin myös laskennan päivitykset. Käyttöliittymän ja toiminnallisuuden toiveet tulevat pääasiassa käyttäjäryhmältä ja tuorein laskentatieto tulee tuotevastaavilta.

2.4. Prosessin haasteet

Yllä kuvattuun prosessiin ja kokonaisuuden rakenteeseen liittyy muutamia isoja haasteita, joiden takia systeemin arkkitehtuurin kehittämiseen on selkeää painetta.

2.4.1. Tarmo2:n haasteet

Tarmo2:n haasteista merkittävimpiä ovat ylläpidolliset haasteet. Nyt koko ohjelman ylläpito on keskitettynä yhdelle instanssille, joka aiheuttaa tälle suhteettoman suuren työkuorman. Tämän takia päivitysaikataulut venyvät pitkiksi, jolloin käytössä on harvoin viimeisin laskentatieto. Toiseksi, pidemmällä tähtäimellä ylläpidon keskittyminen yhdelle instanssille aiheuttaa myös resurssiriskin. Ohjelman päivittyminen ja edelleen kehitys pysähtyisi täysin, jos tämä henkilö olisi syystä tai toisesta pidemmän aikaa poissa tai ei olisi enää lainkaan Metson käytettävissä. Toisaalta tämäntapainen laskentatieto on Metson ydinosaamista, jota ei haluta ulkoistaa talon ulkopuolelle, vaikka sillä saataisiinkin pienennettyä suhteellista resurssiriskiä.

Tarmo2:n ylläpidon keskittyminen yhdelle taholle liittyy ohjelman rakenteeseen. Kun suuri osa kokonaisohjelman toiminnallisuudesta on upotettu laskentamoduulien yhteyteen, se tekee laskentojen päivittämisestä haastavaa. Jotta laskenta- tai toimintapäivityksiä voi tehdä, tulee ohjelman toiminta tuntea melko tarkasti. Tuotevastaavat joutuvat tekemisiin niin monien laskentaohjelmien kanssa, että yksittäisen ohjelman näin läpikotainen tunteminen ei ole perusteltua.

2.4.2. Kokonaisuuden haasteet

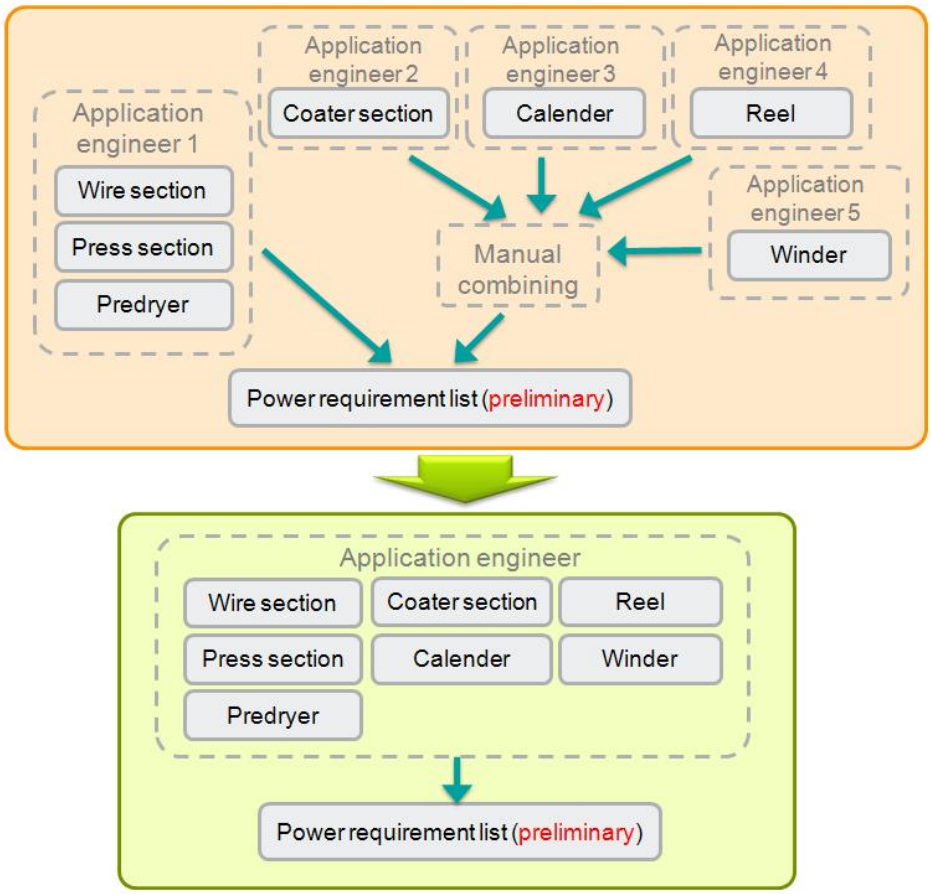
Lisähaasteita tulee myös linjatason integraation takia. Tällä hetkellä suurimmat linjatason haasteet liittyvät siihen, että käytössä ei ole yhtenäistä laskentaprosessia tai toimintamallia. Toisaalta laskennan läpivientiin tarvitaan myös monen henkilön osallistuminen (*kuva 2.6* - ylempi laatikko). Näistä syistä laskennan läpimenoaika on suhteellisen pitkä ja vaihtelee huomattavasti kunkin henkilön työkuorman mukaan. Vanhan sanonnan mukaisesti prosessi on tällä hetkellä yhtä vahva kuin sen kulloinenkin heikoin lenkki. Koska teholaskentaa tehdään myös tarjousvaiheessa, jolloin revisioita tulee lukuisia, tämä kertauttaa myös yksittäisten osaprosessien läpimenoaikojen varianssien aiheuttamia ongelmia.

Nykyinen toimintatapa aiheuttaa myös loppuasiakkaan näkökulmasta haasteita. Tällä hetkellä kun tulokset lasketaan erilaisilla ohjelmilla, ovat myös tuotokset ulkoasultaan hieman erilaisia. Tämä voi asiakkaan näkökulmasta vaikuttaa erikoiselta, koska he kuitenkin kommunikoivat ”Yhden Metson” kanssa.

Kaiken kaikkiaan voikin todeta, että nykyinen linjatason teholaskentaprosessi vaikuttaa orgaanisesti ajan kanssa muodostuneelta. Tämän takia se kärsii kirjavien toimintamallien aiheuttamista vaivoista: epäyhdenmukainen toiminta, epäyhdenmukaiset tuotokset, aikataululliset ongelmat, prosessin hallinnan ja seurannan ongelmat.

2.5. Prosessin kehittäminen

Nykyisessä myynnin organisaatiossa eri Metso Paperin yksiköt toimivat linjamyynnissä rinnakkain saman organisaation alla. Tämä edesauttaa hakemaan synergiaetuja ja antaa hedelmällisen maaperän kehittää toimintaa yhä yhtenäisempään suuntaan niin suuressa mittakaavassa kuin yksittäisissä pienemmissä käytännöissä – kuten mitoitusasioissa, joista nyt puhutaan.



***Kuva 2.6** – Nykyinen ja tuleva prosessi*

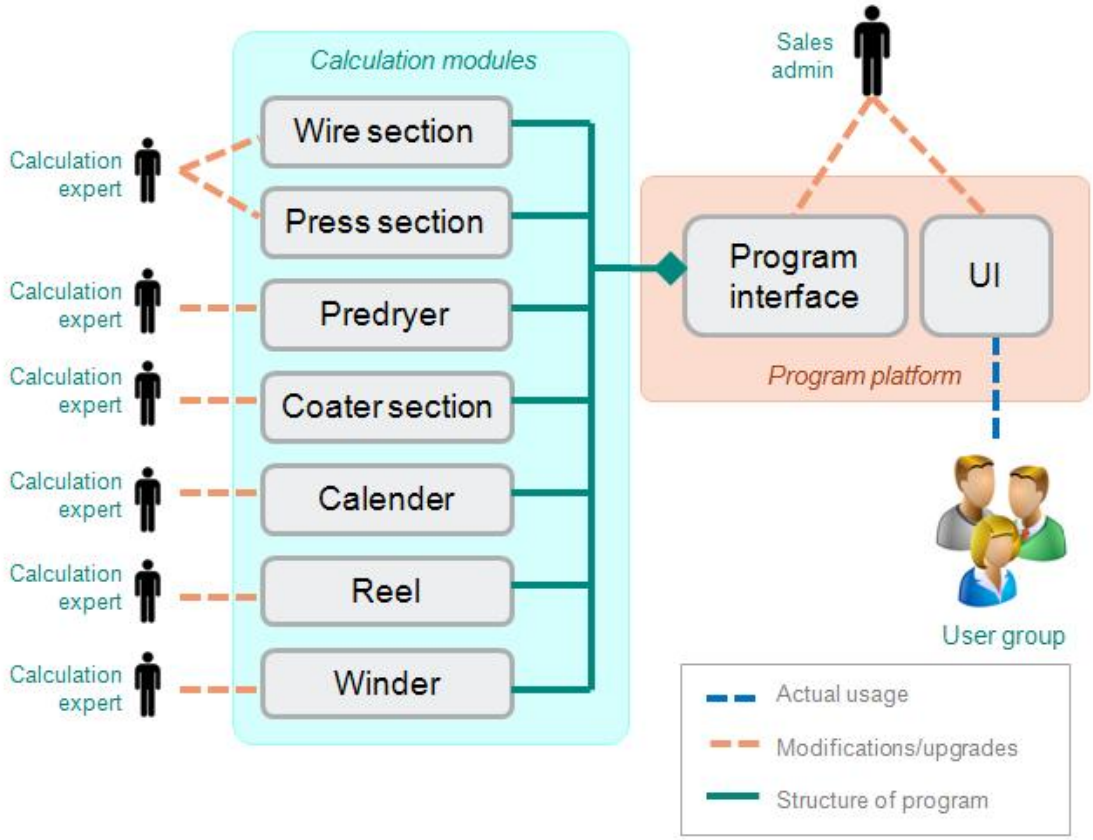
Kuvan 2.6 ylemmässä, oranssissa laatikossa on nähtävissä nykyinen toimintamalli tehontarveluettelon muodostuksessa. Aiemman prosessin haasteiden takia toimintatapa haluttiin rakentaa uudelleen ja *kuvan 2.6* alemmassa, vihreässä laatikossa on esitetty toimintatapa johon prosessi pyritään viemään. Aiempaan prosessiin verrattuna suurimmat erot ovat virtaviivaisuus ja se, että koko mitoitusprosessin läpivienti onnistuu jopa yhden henkilön toimesta. Tässä mallissa kaikki mitoitusperusteet on integroitu samaan ohjelmaan, yhdenmukaiseen käyttöliittymään ja niin helppokäyttöiseen muotoon, että koko mitoitusprosessi on mahdollista viedä läpi, vaikka mitoittaja ei omaisi syvällisiä tietoja jokaisesta osa-alueesta. Tästä syystä ohjelman tulee mahdollistaa myös useamman eri tahon vaikuttaminen laskentaan. Jos tehontarvelaskennan tekevä tarjousinsinööri tai muu käyttäjä on epävarma, jonkun osa-alueen lähtötiedoista tai tuloksista, tulee ohjelman mahdollistaa helpolla tavalla se, että kyseisen osa-alueen asiantuntija pääsee myös laskentaan käsiksi. Tällöin asiantuntija voi tehdä tarvittavat muutokset ja tarkistukset, jotta varmistetaan, että mitoitus on tehty oikein.

Tämä yksinkertaisempi kokonaisprosessi ratkaisee suurimmat aiemman prosessin ongelmat seuraavalla tavalla:

- Pitkä läpimenoaika ja sen varianssi → Yksi ihminen voi tehdä laskennan, jolloin läpimenoaika ja varianssi pienenee → Helpommin hallittava kokonaisuus

- Tarjousvaiheen revisioinnin työläys → Yksi ihminen voi hoitaa revisioinnin, eikä manuaalisia parsimisia tarvita, kun ohjelma tekee tämän automaattisesti
- Dokumenttien yhdenmukaisuusongelma → Laskennat on ohjelman kautta pakotettu yhdenmukaisiksi, jolloin myös lopputuotokset ovat yhdenmukaisia
- Moninaiset toimintamallit → Ohjelman ansiosta kaikki laskennat on samassa kokonaisuudessa ja viedään läpi samalla tavalla

Näiden lisäksi integraalisempi prosessi mahdollistaa parhaiden menetelmien ja laskentamallien tehokkaamman siirtymisen eri osa-alueiden välillä. Ennen mitoitustietoja ja kaavoja on pääasiassa kehitetty toisistaan riippumattomasti, mutta tuleva integraalisempi prosessi mahdollistaa paremmin ristikkäistä oppimista eri alueiden ja toimintojen välillä.



Kuva 2.7 *– Tulevan Tarmo 3:n toiminta*

Jotta *kuvan 2.6* kaltainen prosessi voidaan toteuttaa, täytyy määritellä rakenne, jolla tähän päästään. *Kuvassa 2.7* on kuvattu karkea rakenne, jolla pyritään vastaamaan uuden toimintamallin vaatimuksiin. Tämän rakenteen kehityspohjana on käytetty *kuvassa 2.5* esitettyä Tarmo2:n rakennetta. Kokonaisuuden kasvamisen ja päivitettyjen vaatimusten, sekä haasteiden takia kokonaisrakenne on kuitenkin erilainen kuin Tarmo2:n alkuperäinen rakenne. Uudessa rakennemallissa suurimmat erot ovat käyttöliittymän ja laskennan selkeä erottaminen toisistaan, sekä ylläpidon hajauttaminen.

Käyttöliittymä (UI) ja ohjelmarajapinta (application progamming interface, API) muodostavat pääohjelman. Käyttöliittymä on peruskäyttäjien kanava laskentaan ja ohjelmarajapinta on kanava käyttöliittymän ja laskennan välillä. Laskenta on puolestaan hajotettu laiteryhmien kokoisiin palasiin, jotka kukin sisältävät kyseisen rakenneryhmän laskentakaavat. Tarkempi kuvaus arkkitehtuurista löytyy kappaleesta *6. Tarmo 3.0*.

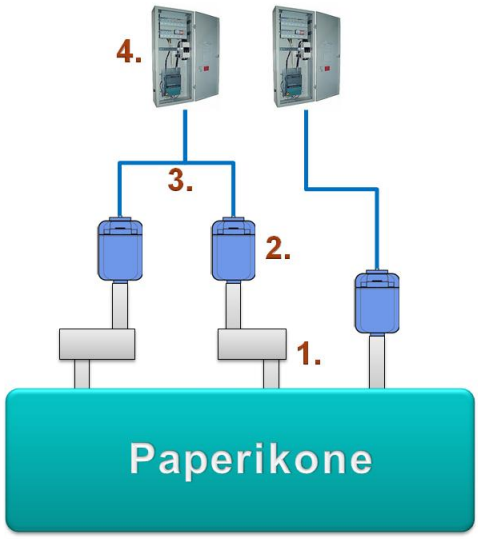
Ylläpidon hajauttaminen tapahtuisi puolestaan siten, että myynti ottaisi linjatason toimintona vastuun pääohjelman, eli käyttöliittymän ja ohjelmarajapinnan ylläpidosta. Tällöin yhteinen linja pysyy läpi koko käyttöliittymän ja ohjelman sisäisen toiminnan, jolloin käyttöliittymän parannukset voidaan helposti implementoida koskemaan kaikkia osa-alueita. Tämä lähestyminen mahdollistaa myös sen, että ohjelmallista toimintaa, käyttöliittymää tai vastaavia kokonaisuuksia ei tarvitse sekoittaa laskentapohjiin. Tämä tekee laskentapohjista yksinkertaisempia. Laskentamoduuleihin tulee pääohjelman takia ainoastaan input-/output-rajapinta, jonka kautta tieto liikkuu pääohjelman ohjelmarajapinnan kautta käyttäjältä ja käyttäjälle. Muutoin laskentamoduulit voidaan toteuttaa siten miten kunkin moduulin ylläpitäjä näkee parhaaksi. Tämä mahdollistaa tehokkaan vanhojen laskentapohjien käytön, eikä kaikkia laskentoja tarvitse alkaa rakentamaan tyhjältä pohjalta.

Kokonaisjärjestelmä on pyritty rakentamaan siten, että se vaatisi kaikilta osapuolilta mahdollisimman vähän työtä ja uuden oppimista:

- Vaikka kokonaisarkkitehtuuri onkin uudenlainen, niin kokonaisuus on pyritty rakentamaan selkeäksi ja hyvin kommentoiduksi, joka helpottaa myynnin ylläpitäjän omaksumista.
- Laskentamoduulien sisäinen arkkitehtuuri on vapaa, joten jokainen ylläpitäjä voi valita haluamansa laskenta-arkkitehtuurin ja hyödyntää vanhoja laskentoja halutessaan täysmääräisesti.
- Käyttöliittymä on rakennettu Tarmo2:n mukaisesti, tehden siihen vain pieniä parannuksia. Näin uuden Tarmon käyttöönotto vanhoille käyttäjille on hyvin kivuton ja tietyllä tapaa huomaamaton prosessi.
- Muiden laskentapohjien käyttäjät joutuvat sen sijaan totuttautumaan uuteen järjestelmään ja ulkoasuun. Kyseistä prosessia on kuitenkin testattu ja hiottu paremmaksi jo pitkään, joten uudet käyttäjät pääsevät käyttämään pitkälle kehitettyä järjestelmää.

3. TEORIA – PAPERIKONEEN LINJAKÄYTÖT

Paperikoneen linjakäyttötehojen määritys vaikuttaa suureen määrään eri käyttölaitteita suoraan ja epäsuorasti. *Kuvassa 3.1* on esitetty käyttölaitteet joihin linjakäyttöjen mitoitus primääristi vaikuttaa. Kytkimet, vaihteet ja näiden perustukset määräytyvät suoraan käyttötehovaatimusten, sekä telan ja moottorin pyörimisnopeuden perusteella. Moottorit yhtälailla määräytyvät suoraan käyttötehotarpeen mukaisesti riippuen kuitenkin vaihteen välityksistä. Moottorien vaatima sähkökaapeloinnin vahvuus riippuu taas moottorien koosta – kaapelien poikkipinta-ala voi olla muutamista neliömilleistä jopa kymmeniin neliösentteihin. Ohjauslaitteet riippuvat moottorin tyypistä, sekä prosessin asettamista vaatimuksista.



1. Kytkimet ja vaihteet, sekä näiden perustukset
2. Sähkömoottorit, sekä näiden perustukset
3. Sähkökaapelointi
4. Ohjauslaitteet

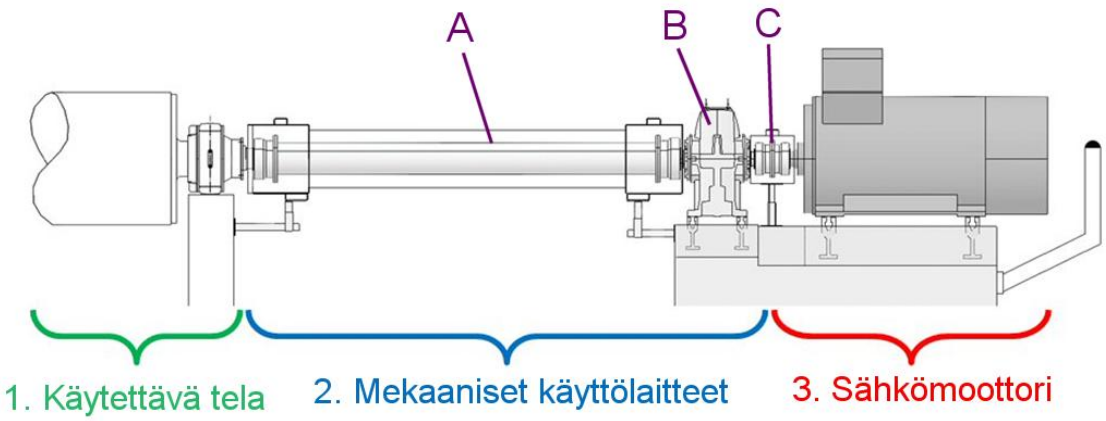
***Kuva 3.1** – Laitteet joihin käyttöpisteiden tehotiedot vaikuttavat*

Edellä esitettyjen asioiden lisäksi tehontarvelaskennan pohjalta tehtävät laitevalinnat ja käyttöjen mitoitus vaikuttaa myös ilmastointikanavien, vedenpoistokanavien ja tehdashallin pystypalkistojen suunnitteluun, koska kaikki nämä kokonaisuudet vaativat huomattavia määriä tilaa, jolloin kokonaisuuden tilankäytöllinen suunnittelu on usein haasteellista.

3.1. Käyttöpisteiden tehotiedot

Paperikoneen käyttöketju koostuu tyypillisesti kahdesta osasta: sähkökäytöistä ja mekaanisista käyttölaitteista. Sähkökäytöllä (k*uva 3.2 – kohta 3.*) tarkoitetaan moottoria (ja sen ohjausjärjestelmiä), joka muuntaa sähköenergian mekaaniseksi liike-energiaksi.

Mekaanisella käytöllä (k*uva 3.2 – kohta 2.*) tarkoitetaan laitteita, jotka välittävät sähkömoottorin tuottaman mekaanisen energian paperikoneen telalle tai sylinterille (*kuva 3.2 – kohta 1.*) ja näin ollen hyötyenergiaksi prosessin näkökulmasta. Kuvassa on merkittynä myös mekaanisen käytön tyypillisimmät laitteet: A = Pitkä kytkin, B = Hammasvaihde, C = Lyhyt kytkin. Tänä päivänä löytyy myös suorakäytettyjä teloja, jolloin sähkömoottori on kytketty suoraan koneen runkoon, eikä telan ja sähkömoottorin välissä ole ylimääräisiä vaihteita tai kytkimiä. Suorakäytöt ovat kuitenkin suhteessa kalliita, eivätkä sovi (ainakaan toistaiseksi) kaikkiin positioihin.



***Kuva 3.2** – Paperikoneen käyttöketju*

Käyttölaitteiden mitoitus tapahtuu käytettävän telan tai sylinterin akselimomentin vaatiman tehontarpeen perusteella. Jokaiselle käyttöpisteelle määritellään kaksi erilaista tehontarvelukemaa: NRL, RDC. Näiden lisäksi määritellään usein myös ”Start”- ja ”Stop”-tehot. NRL:llä (Normal running load) tarkoitetaan normaalia prosessin ajotilanteen vaatimaa tehontarvetta, jossa paperikone-/kartonkilinjalla ollaan saavutettu haluttu vakionopeus. RDC:llä (Recommended Drive Capasity) tarkoitetaan poikkeuksellista kuormitustilannetta, joka on luonteeltaan tilapäinen. Tällaisia poikkeuksellisia tilanteita voivat olla muun muassa kiihdytykset, normaali suuremmat alipainetasot, viivakuormat tai kuivatussylinterin vesilasti [Ketolainen 2002]. RDC:llä pyritään varmistamaan, että käytöillä on tehoreserviä riittävästi, etteivät yllämainitun kaltaiset poikkeustilanteet aiheuttaisi häiriöitä prosessin toiminnassa. RDC on usein tärkeässä roolissa käyttölaitteita mitoittaessa.

PRESS SECTION - OptiPress

Linear loads 1.p 150 kN/m 2.p 800 kN/m S.p 100 kN/m Separate Press 100 kN/m

POS.	ROLL / DRIVE POINT	MACHINE DATA			POWER REQUIREMENTS														GEAR DATA		MOTOR DATA				NOTE
		ROLL	n2	INTERNA	NRL	NRL	RDC	RDC	START	START	ACC	ACC	STOP	STOP	STOP	Q-STOP	Q-STOP	EXTRA	EXTRA	GEAR	GEAR	MOTOR	MOTOR	n1	ROTATI	Rev.
		[mm]	[1/min]	i	[kW]	[Nm]	[kW]	[Nm]	[kW]	[Nm]	[kW]	[s]	[kW]	[Nm]	[s]	[kW]	[s]	[kW]	[s]		i		kW	[1/min]	C/CC	
	1st press																									
	1st press top roll	1515	191		295		392												HGUS				*)			
	1st press bottom roll	1515	191		220		276												HG							
	2nd press																									
	2nd press bottom roll	1220	237		522		676												RG							
	Separate press																									
	Paper guide roll	514	564		5		6												HG							
	Separate press top roll	1025	283		31		39												HGUS							
	Separate press bottom roll	1240	234		40		50												HG							
	Smoothing press																									
	Smoothing press top roll	1025	283		42		63												HGUS							
	Smoothing press bottom roll	1240	234		42		63												HG							

Note
1) Optional. Special motor, included to Metso delivery (inverter not included)
2) Special motor, included to Metso delivery (inverter not included)

***Kuva 3.3** – Esimerkki puristinosan tehontarveluettelosta*

”Start”-teho puolestaan kertoo millaisen tehon käyttöpiste vaatii koneen käynnistyessä. ”Stop”-teho kertoo kuinka paljon tehoa käyttöpisteeltä vaaditaan prosessin pysäyttämiseen. Tämä teho on negatiivinen, jolloin esimerkiksi rullaimella pysäytyksessä käytettävä teho voidaan osittain ottaa talteen ja hyödyntää uuden telan ”starttaamisessa”. Esimerkki asiakkaalle toimitettavasta tehontarveluettelosta näkyy *kuvassa 3.3*.

3.2. Paperikoneen rakenne ja käyttöryhmät

Paperikonetta käyttävät sähkömoottorit jaetaan käyttöryhmiin siten, että moottorit joiden käyttämien telojen kehänopeudet ovat yhtä suuret muodostavat yhden käyttöryhmän [Isokääntä 2007]. Märän pään käyttöryhmiä ovat viiraryhmä, puristinryhmä sekä kuivatusryhmät. Kuivan pään käyttöryhmiä sen sijaan ovat: päällystysryhmä, kalanteriryhmä, rullaimet ja leikkurit.

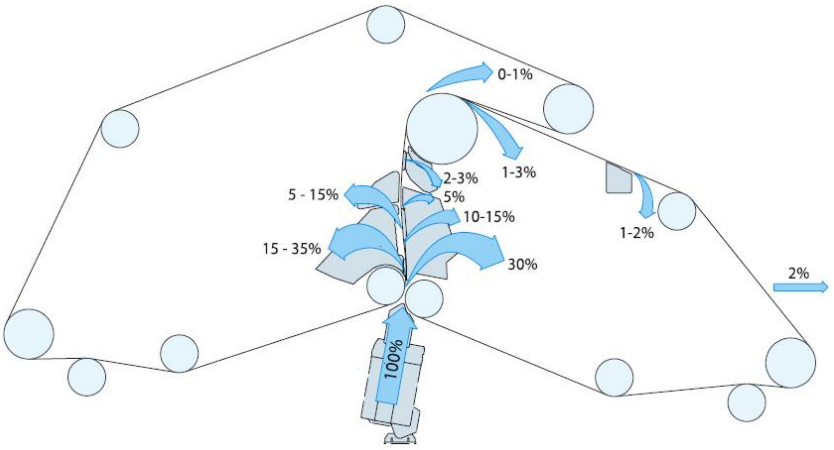
Viira-, ja kuivatusosalla pääosa voimansiirrosta tapahtuu kudosten välityksellä telalta toiselle. Puristimella teho siirtyy kudosten lisäksi myös nippien kautta. Kalanterilla pääosa tehoista siirtyy nippien lisäksi paperiradan itsensä välityksellä. Päällystysasemilla taas lähes kaikkien telojen tulee olla erilliskäytettyjä, koska päällystysvaiheessa prosessia ei käytetä enää kudoksia, jotka siirtäisivät tehoa telalta toiselle.

Kussakin käyttöryhmässä on yleensä useampia käyttöjä ja niiden määrä riippuu koneen rakenteesta, koosta ja välitettävien voimien suuruudesta. Nyrkkisääntönä voikin todeta, että mitä suurempia tehoja pitää välittää, sitä enemmän käyttöpisteitä on. Vaikka olisikin teknisesti mahdollista tuottaa kaikki tarvittava käyttöteho yhdestä pisteestä, ei tätä voimaa saada yleensä järkevästi välitettyä koko kudoskiertoon, koska voima siirretään kudoksen ja telan välisen kitkan välityksellä, joka rajoittaa yhden telan välittämää maksimitehoa. Paperikoneesta löytyy pääkäyttöjen lisäksi yleensä myös ryömimiskäyttöjä, joita käytetään esimerkiksi prosessin ylösajon aikana.

Paperikoneen kuivassa päässä käyttöpisteitä on tyypillisesti enemmän, koska voimia ei voida siirtää kudosten välityksellä. Paperiradan välityksellä voimia ei voida siirtää, koska se aiheuttaisi huomattavan riskin ratakatkojen lisääntymiseen. Näin ollen lähes kaikki telat ovat itsenäisesti käytettyjä tai mahdollinen voimansiirto telojen välillä tapahtuu pääasiassa telojen välisten nippien kautta (esimerkiksi kalanterilla).

3.2.1. Viiraosa

Viiraosan tehtävänä on poistaa perälaatikolta tulevasta massasuspensiosta vettä siten, että muodostuu vaaditut ominaisuudet täyttävä paperiraina. Hallitun rainanmuodostuksen lisäksi viiraosan tehtävä on saattaa raina riittävän korkeaan kuiva-ainepitoisuuteen (*kuva 3.4*), jotta sen siirto viiralta puristimelle on helppoa ja puristinosalla saavutetaan hyvä ajettavuus. [Paulapuro 2000]

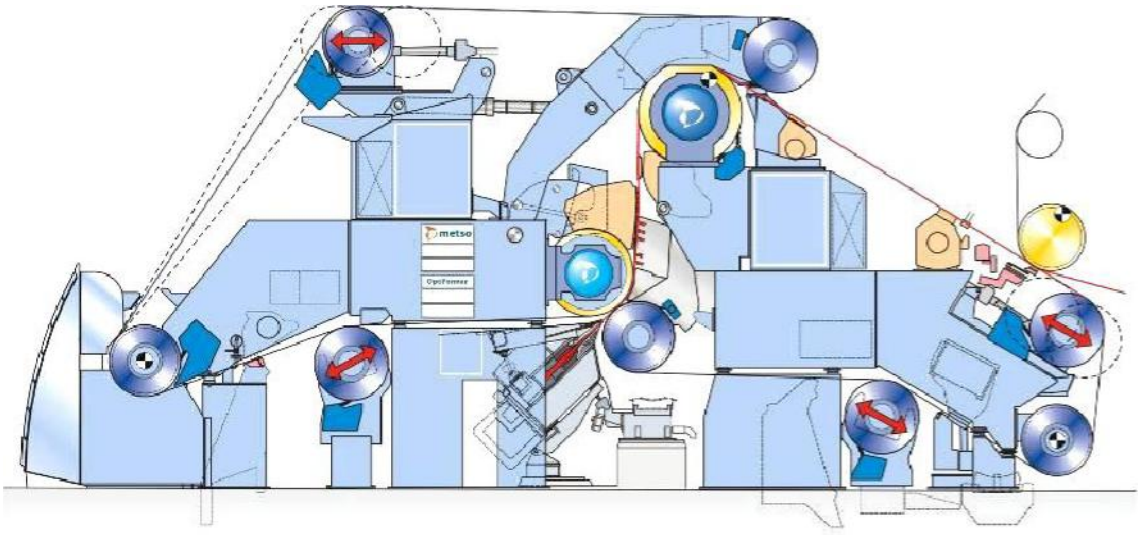


***Kuva 3.4** – Metson "BelBaie V"-formerin vedenpoiston esimerkkikaavio [Metso Prodcut Vault 2011]*

Vanhin viiraosan tyyppi on tasoviira, jossa veden poisto tapahtuu yhteen suuntaan tasomaisen viiran läpi. Hybridiformerissa on tasoviiran jatkeeksi tehty yläpuolinen viiraosa, jonka avulla saavutetaan parempi vedenpoiston symmetria. Tasoviiroja ei enää juurikaan valmisteta, mutta niitä on vielä jonkin verran käytössä. Hybridiformereita käytetään uusinnoissa, sekä hitaammilla koneilla ja paksummilla paperilajeilla. [Paulapuro 2000]

Uusissa, nopeissa paperikoneissa yleisin viiraosan rakenne on kitaformeri. Kitaformerissa massasuspensio tuodaan kahden viiran väliseen kitaan, jossa vedenpoisto tapahtuu alusta asti molempiin suuntiin (*kuva 2.3*). Näin ollen vedenpoisto on perinteistä tasoviiraa symmetrisempi ja näin ollen rainan ajettavuus huomattavasti parempi. *Kuvassa 3.5* on esitetty Metson käyttämä OptiFormer-tyypin kitaformerirakenne.

Nykyään yhä tärkeämmän roolin metson portfoliossa saavat kartonkikoneet ovat nopeudeltaan paperikoneita hitaampia, mutta niiden massavirrat suhteessa nopeuteen huomattavan suuria ja tämä asettaa viiraosalle erilaisia haasteita. Siksi kartonkikoneiden viiraosat poikkeavat rakenteeltaan jokseenkin paperikoneista. Kartonkikoneilla suositaan esimerkiksi monikerros-viiraosaa tai normaalia suurempiin vedenpoistoihin kykeneviä hybridiformereita.



***Kuva 3.5** - Metso OptiFormer-kitaformeri. [Metso Prodcut Vault 2011]*

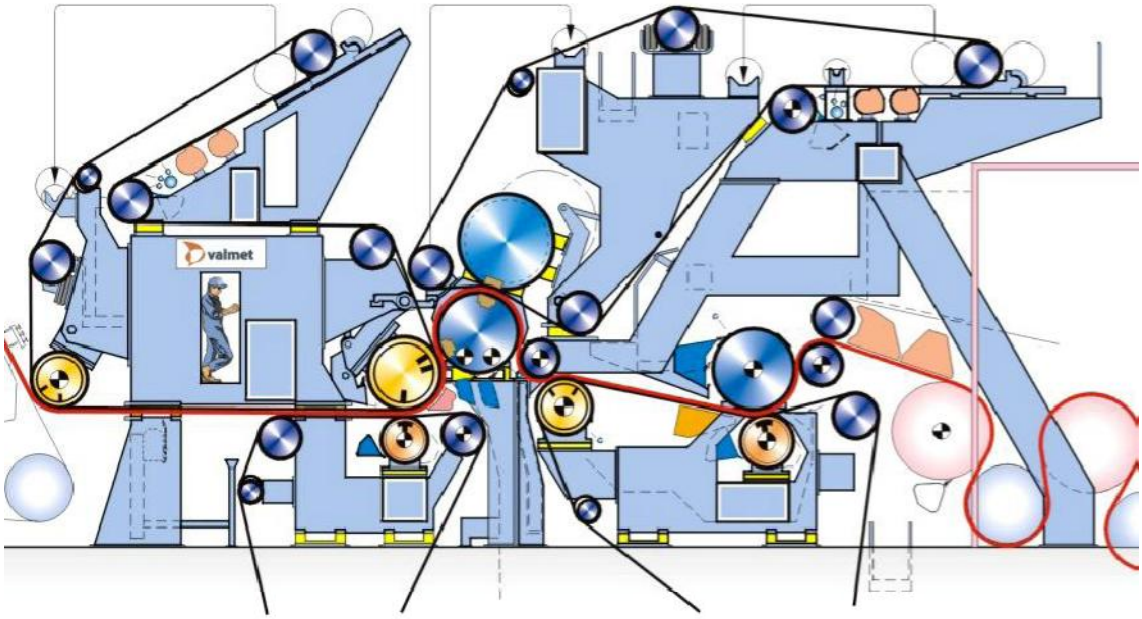
Viiraosan käytettyjä teloja ovat yleensä viirojen vetotelat, sekä ainakin yksi imutela (kuvassa käyttöpisteet on esitetty ”◓”- merkinnällä). *Kuvan 3.5* Optiformerissa käytettyjä teloja ovat ulko- ja sisäviiran vetotelat sekä viiranimutela. Loput telat pyörivät viiravetoisina käytettyjen telojen määräämällä nopeudella. Paperikoneen viiraosalla suurin osa käyttötehosta kuluu viiran ja paikallaan pysyvien vedenpoisto-osien (kaapimet, imurit, yms.) väliseen kitkaan. Viiraosalla joudutaan suuren vedenpoistotarpeen takia käyttämään suuria alipaineita, jotka aiheuttavat huomattavaa laahauskitkaa viiran ja vedenpoistokalusteiden välillä. Muita tehokaivoja ovat telojen sisäiset deformaatiot ja kitkat, kaapimien aiheuttama laahauskitka, viirojen muodonmuutokset, sekä nippien aiheuttamat tehohäviöt. Viiraosan voimansiirtoketjussa kriittisin kohta on yleensä voimansiirto käytetystä telasta viiraan, jolloin laskennallisesti kriittinen kohta on telojen ja kudosten välisen tehonsiirtokyvyn laskenta. [Karjalainen 1998]

3.2.2. Puristinosa

Puristinosan tehtävänä on jatkaa viiraosan aloittamaa vedenpoistoa ja vaikuttaa paperin rakenteellisiin ominaisuuksiin. Puristimella vedenpoiston filosofia on viiraosaan verrattuna erilainen ja se tapahtuu paperirainasta mekaanisesti puristamalla, sekä puristinhuopiin siirtämällä, josta siirtynyt vesi imetään huopaimureilla pois. Mekaaninen puristaminen tapahtuu kahden toisiaan vasten olevan telan, sekä huopien välisessä nipissä. Puristinosan tavoitteena on poistaa rainasta mahdollisimman paljon vettä ja tiivistää sitä. Käyttöjen taloudellisuus ja hyvä ajettavuus korostuvat puristinosalla, sillä linjakäyttöjen näkökulmasta puristimen tehontarpeet ovat paperikoneen suurimmat. [Karjalainen 1998]

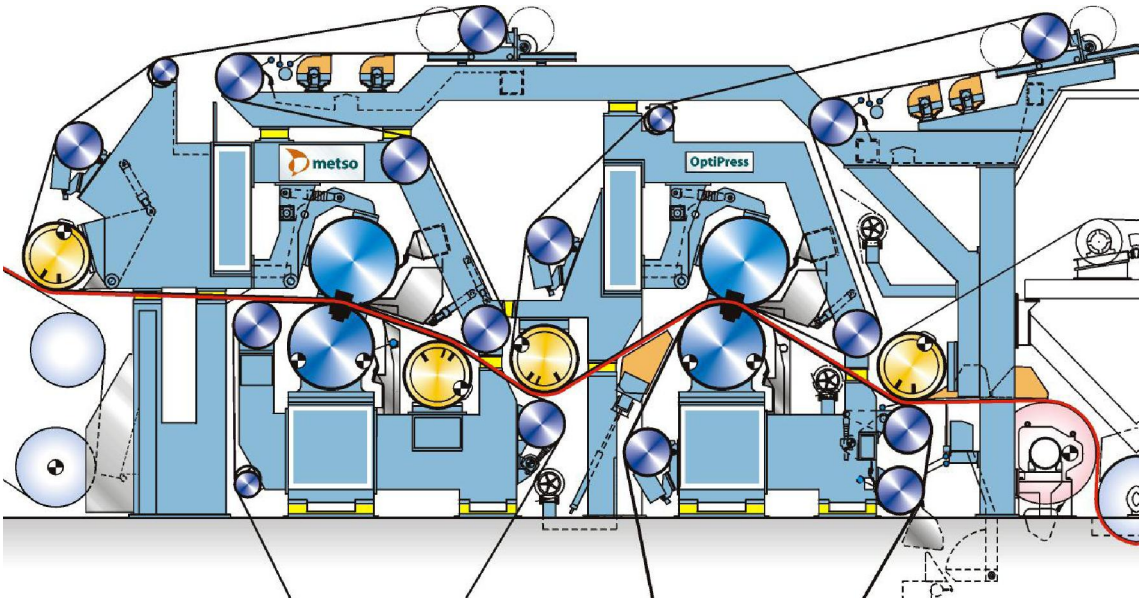
Valmistettavan paperilajin laatu ja kuiva-ainepitoisuus vaikuttavat puristinkonseptin valintaan, joka määrää rakenneryhmän rakenteen ja puristimen nippien määrän. Puristimesta on kaksi tyypillisintä rakennekonseptia: keskitelallinen puristin ja suora puristin.

Kuvassa 3.6 on keskitelallinen puristin. Keskitelallisessa puristimessa paperiraina ajetaan neljän telan ja kolmen puristinnipin läpi. Tyypillisessä keskitelakonseptissa on taipumakompensoitu tela, imutela, solid-tela ja belt-tela. Näin kyseistä puristinkonseptista löytyy kaksi lyhyttä telanippiä, sekä yksi pitkä ”kenkänippi”. Joskus käytetään vielä neljättä erillispuristinta, jonka avulla saadaan kuivattua ja muokattua rainaa vielä enemmän. Tässä konseptissa käyttöpisteinä toimivat puristimille rainan tuova pickup-, taipumakompensoitu- ja keskitela, sekä osa siirtoteloista riippuen koneen tehontarpeesta ja käyttöpisteiden tehonsiirtokyvystä. Kuvassa esimerkkikonseptin käyttöpisteet on esitetty ”◓”- merkinnällä. Suurin käyttämätön tela on pitkän puristusnipin ylätelana toimiva kenkätela. Se ei rakenteensa vuoksi kykene siirtämään rataan käyttövoimaa ja toimii siksi sekundaarisena telana, jolle tarvittava liike välittyy nipin kautta. [Pirinen 2005]



***Kuva 3.6** - Metson keskitelapuristin varustettuna erillispuristemella [Metso Prodcut Vault 2011]*

Tuoreempi puristinkonsepti on ”suora puristin” (straight through press), joka löytyy Metsolta konseptinimellä OptiPress (*kuva 3.7*). Suorassa puristimessa jokainen nippi on toteutettu kahden erillisen telan avulla. Tämä mahdollistaa nippien yksilöllisen säädön ja näinollen myös puristimen kokonaisprosessin säädettävyys paranee. Konseptin ansiosta myös rainan ajo on ”suorempi” eli raina tekee sivusta katsottuna vähemmän mutkia. Tämä mahdollistaa hieman paremman rainan rakenteen, kun puristimella yhä huokoinen paperiraina ei joudu mutkien aiheuttamien rasitusten kohteeksi. Suorassa puristimessa nippejä voi olla yhdestä kolmeen ja nipit voidaan toteuttaa hyvin erilaisten telojen välissä riippuen siitä, mitä kultakin nipiltä haetaan: hyvää bulkkia, hyvää puristusvaikutusta, pinnan sileyttä tms. [Pirinen 2005] Opti-konseptissa käytetään tyypillisesti ensisijaisesti pitkiä nippejä.



***Kuva 3.7** - Metson OptiPress-puristin kahdella kenkänipillä [Metso Prodcut Vault 2011]*

Kenkätelan rakenne suosii öljykalvovaatimuksensa takia ylätelapositiota, minkä takia suurin osa Opti-konseptin tehontuotosta pitää tapahtua alaviiran puolelta, josta suurin osa yläviirakierron vaatimasta tehosta siirretään nippien kautta. Yläviiroillekin sijoitetaan joitakin käyttöjä – yleensä imuteloille, joilla on käyttöjen kannalta tärkeä, pitkä kontaktialue viiran kanssa. Alakierroilla käytöt on yleensä sijoitettu kenkätelojen koville vastateloille, sekä imuteloille ja suuremmilla koneilla joskus myös viiranjohtoteloilla.

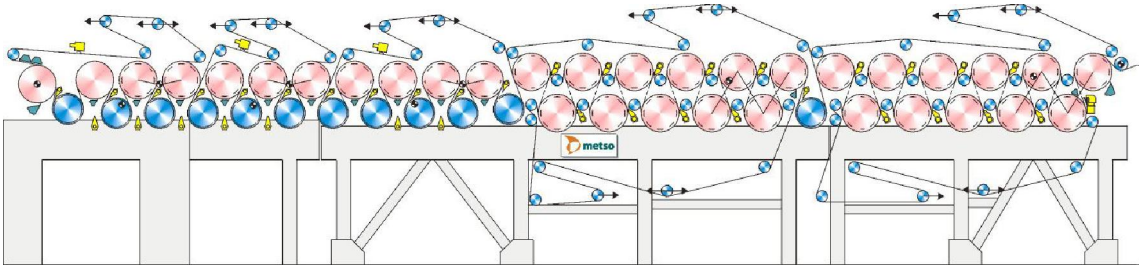
Käyttöjen kannalta merkittävä ero suoran ja keskitelallisen konseptin välillä on se, että keskitelallisessa konseptissa käytöt pakkautuvat huomattavan tiiviille alueelle, joka tekee käyttöjen sijoittelusta haastavaa, koska käytöt ovat isoja ja vaativat tukevat perustukset. Moottorit pitäisi tämän lisäksi sijoittaa mahdollisimman lähelle konetta, jotta välitysakselit eivät veny liian pitkiksi, koska tämä aiheuttaa värähtelyherkkyyttä. Tämä aiheuttaa tehdassuunnittelulle suuria haasteita, koska moottoreiden ja välitysten lisäksi koneen käyttöpuolelle pitää sijoittaa huomattavia määriä muitakin tilaa vieviä laitteita ja kanavistoja. Tämän takia suoran puristimen käyttöjen ja muiden järjestelmien suunnittelu on tehdassuunnittelun näkökulmasta helpompaa. [Isokääntä 2007]

3.2.3. Kuivatusosa

Kuivatusosan tarkoitus on kuivattaa paperiraina linjan kokonaiskonseptista riippuen yli 90% kuiva-ainepitoisuuteen (KAP). Rainaa kuivataan höyrystämällä siinä yhä olevaa vettä pois. Tyypillisin ja energiatehokkain kuivatustapa on sylinterikuivatus, jossa paperiraina painautuu kuumaa sylinerin pintaa vasten, joka lämmittää rataa ja höyrystää siitä kosteutta. [Niskanen 1997]

Perinteissä sylinterikuivaukseen perustuvissa kuivatusosissa on kaksi pääkonseptia: yksi- ja kaksiviiravienti. Näistä perinteisempi on kaksiviiravienti (k*uva 3.8*),

missä on ainoastaan kuivatussylintereitä. Kuivatussylintereiden suuri määrä mahdollistaa hyvän kuivatustehon suhteessa linjan pituuteen. Prosessin ajettavuus ja paperirainan kosteusprofiilin homogeenisyys eivät sen sijaan ole hyvällä tasolla, koska vapaat viennit, paperirainan konesuunnan kutistuminen ja taskujen tuuletus eivät ole yhtä helposti hallittavissa kuin yksiviiraviennissä. Toisaalta myös kaksiviiraviennin mukaisen kuivatuskonseptin ylösajossa käytettävä köydellä tehtävä päänvienti on vanhanaikainen menetelmä, joka vaatii omat käyttönsä. [Paulapuro 2000]

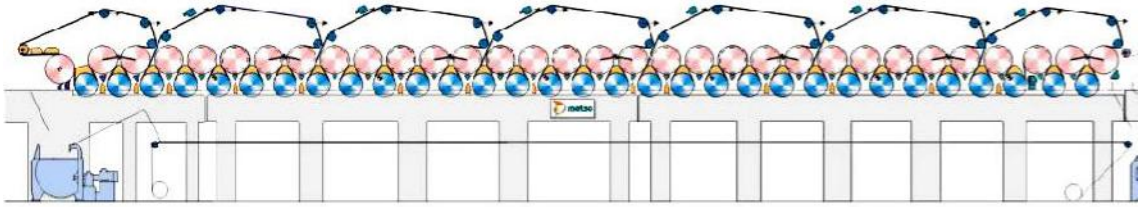


***Kuva 3.8** - Metson SymDry-kuivatusosa, jossa alkuosassa yksiviiravienti ja loppuosassa Kaksiviiravienti [Metso Prodcut Vault 2011]*

Kaksiviiraviennissä käytöt pitää sijoittaa kumpaankin viiralenkkiin, jotta voidaan varmistua riittävästä tehosta kaikille teloille ja sylintereille. Nopeuseron kahden viiran välillä tulee olla hyvin pieni, jotta vältytään ratakatkoilta ja muilta ajettavuusongelmilta. Tämän takia ylä- ja alarivin käytetyt sylinterit sidotaan usein toisiinsa mekaanisten hammasvaihteiden kautta, jolloin kaikkia sidottuja teloja käytetään samassa tahdissa. Nykyään säätöjärjestelmien tarkentuessa on kuitenkin tullut mahdolliseksi myös kummankin huopalenkin itsenäinen käyttö, jolloin käyttöjä on joissain projekteissa sijoitettu sylintereiden lisäksi myös huopien omille ohjainteloille.

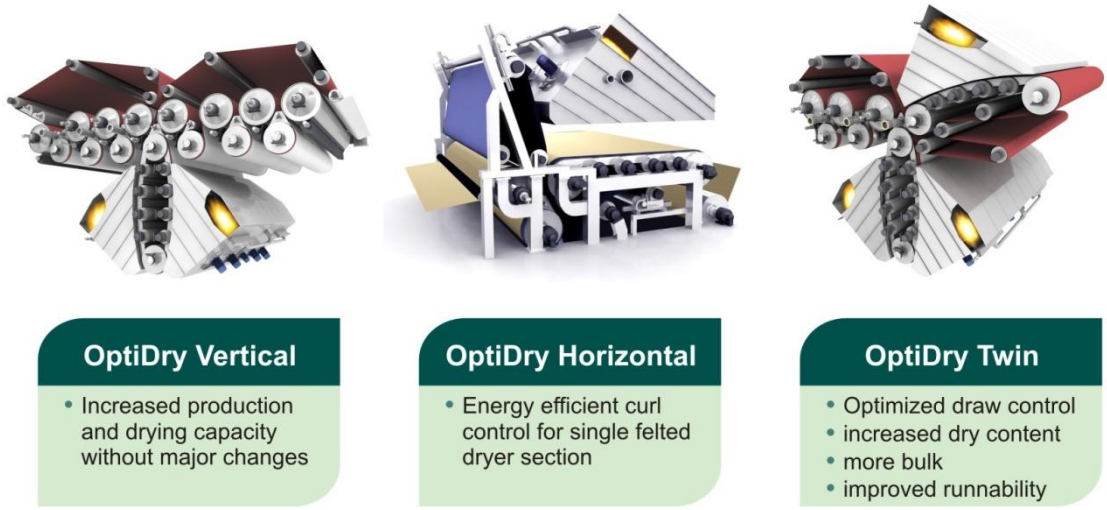
Tänä päivänä yleisempi kuivatusosan konsepti on yksiviiravienti (*kuva 3.9*), jossa höyrysylintereitä on vain yhdessä rivissä ja näiden parina on VAC-teloja. VAC-telat ovat uritettuja ja pienillä porauksilla varustettuja imuteloja. Tämä mahdollistaa täysin suljetut viennit, jolloin prosessin ajettavuus on huomattavasti parempi kuin kaksiviiraviennissä. Yksiviiravienti mahdollistaa myös paremman, köydettömän, puhalluksin toimivan päänviennin. [Paulapuro 2000]

Yksiviiravientikonseptissa jokaisessa kuivatusryhmässä on tyypillisesti kolme käytettyä telaa: kaksi sylinteriä ja yksi VAC-tela. Sylinterikäytöt ovat yleensä yhdistetyn vaihteen takana yhden moottorin pyöritettävänä ja VAC-telalla on oma erilliskäyttönsä. Muille teloille ja sylintereille käyttöteho välittyy viiran kautta. Päällystyksen (coating) jälkeen tilanne on kuitenkin toinen, koska paperiradan ja sylinterien välinen kitka on pienempi kuin etukuivatusosalla. Tämän takia tehoja ei pystytä välittämään radan/viiran avulla ja näin ollen kaikki telat ovat käytettyjä. Kuivatusryhmien loppupuolella sijaitsevien peruskäyttöjen lisäksi kuivatusosan ensimmäisellä telalla on erilliskäyttö, jonka tehtävä on ensisijaisesti ylläpitää puristimen ja kuivatusryhmän välistä kireyseroa. [Isokääntä 2007]



Kuva 3.9 *- Metson SymRun-kuivatusosa, jossa yksiviiravienti [Metso Prodcut Vault 2011]*

Nykyään kuivatusosalle on tullut myös muutamia uusia laitteita – OptiDry Vertical ja OptiDry Horizontal (*kuva 3.10*), jotka ovat ns. päällepuhalluskuivaimia. Näitä laitteita ei käytetä ainoana laitteena kuivatusosalla, vaan lähinnä paperin profilointiin tai kuivatusosan tuotantokapasiteetin nostoa ja kokonaislinjan pituuden suhteellista laskua varten. Energiatehokkuudeltaan OptiDry-konseptit eivät ole sylinterikuivatuksen tasolla, mutta absoluuttiselta kuivatusteholtaan ne ovat tehokkaampia ja kompaktimpia. Tämän ansiosta niitä asennetaan vanhoihin linjoihin, jotta kuivatuskapasiteetin pullonkaulaa saataisiin hieman avarrettua.



Kuva 3.10 *– Päällepuhalluskuivaimet [Metso Prodcut Vault 2011]*

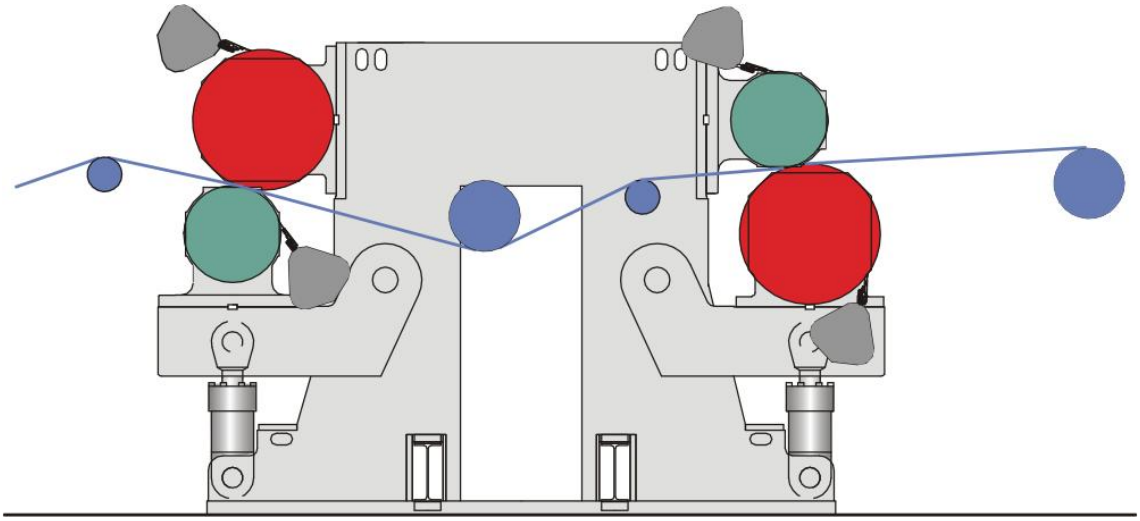
Käyttöjen kannalta kuivatusosa aiheuttaa haasteita kuuman ja kostean ympäristön takia. Toisaalta taas tilankäyttö- ja sijoitussuunnittelu on huomattavasti helpompaa kuin paperikoneen märässä päässä, koska käyttöpuolella on suhteessa vähemmän käyttöjä ja muita toimilaitteita. Nykyään kuivatusosalla käytetään myös runkoon kiinnitettäviä suorakäyttömoottoreita, jolloin vältytään vaihteilta ja pitkiltä kytkimiltä, mikä pienentää kuivatusosan investointikustannuksia perustusten ja varaosien osalta. Suorakäyttömoottorit ovat kuitenkin kalliita ja prosessiolosuhteiden haasteet kärjistyvät kun moottorit ovat huuvan sisällä. Tästä syystä perinteisemmät, vaihteelliset käytöt ovat yhä suosiossa.

Kuivatusosalla on pituutensa ansiosta myös muita haasteita käyttöjen kannalta. Pitkässä kuivatuslinjassa käyttötehon säätö on haastavaa ja usein käy niin, että viimei-

sen kuivatusryhmän käytöille kertyy suurempi tehokuorma, kun se pyrkii ylläpitämään radan kireyttä, jota on periytynyt koko radan matkalta. Viimeisten/viimeisen ryhmän käytöt käyvät verrattain korkealla käyttöteholla suhteessa alkupään käyttöryhmien normaalin ajotilanteen kuormitukseen.

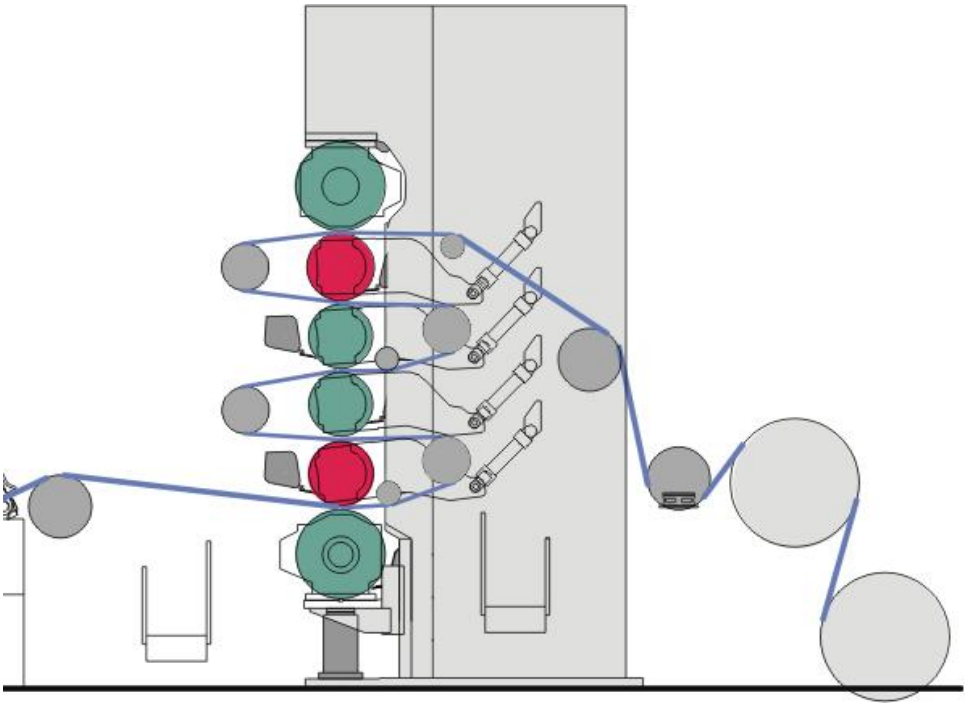
3.2.4. Kalanteri

Kalanteri kuuluu paperikoneen kuivan pään laitteisiin. Kalanterin tehtävä on parantaa paperin pinnanlaatua, kiiltoa ja vaikuttaa paperin bulkkiin halutulla tavalla. Kalantereita on kahta päätyyppiä: yksi- ja moninippisiä. Yksinippisissä kalantereissa (k*uva 3.11*) on nimensä mukaisesti vain yksi tai joissain erikoistapauksissa kaksi puristusnippiä. Puristusnippi on tyypillisesti pitkä, jolloin toinen teloista on pinnoitettu pehmeämmällä materiaalilla. Tämän takia paperin puolet eivät tasoitu identtisesti. Metsolla yksinippisiä kalantereita löytyy mm. tuotenimillä OptiSoft ja OptiHard. Niitä käytetään laitteen positiosta riippuen muun muassa seuraaville paperilajeille: sanomalehtipaperi, hienopaperit, kartongit, sekä aikakausilehtipaperit.



***Kuva 3.11** - Kaksi nippinen OptiSoft-kalanteri [Metso Prodcut Vault 2011]*

Moninippisessä kalanterissa (*kuva 3.12*) - kutsutaan myös superkalanteriksi - nippejä on neljästä jopa kahteentoista. Moninippinen kalanteri on yleensä yhdessä tai kahdessa korkeassa telapinossa (roll stack). Kunkin moninippikalanterin ensimmäinen ja viimeinen tela on tyypillisesti pitkänippinen SYM-tela. Välitelat ovat vuorotellen termo- ja polymeeriteloja. Tämän ansiosta myös välinipit muokkaavat paperiradan rakennetta tehokkaasti keskipitkien nippien ja lämmön yhteisvaikutuksen ansiosta. Moninippi-kalantereita käytetään, kun halutaan paperilta erityisen voimakasta pintakiiltoa. Superkalanteroituja paperilaatuja voivat olla esimerkiksi kiiltäväpintaiset hienopaperit ja aikakausilehtipaperit.



***Kuva 3.12** – OptiLoad 6-kalanteri [Metso Prodcut Vault]*

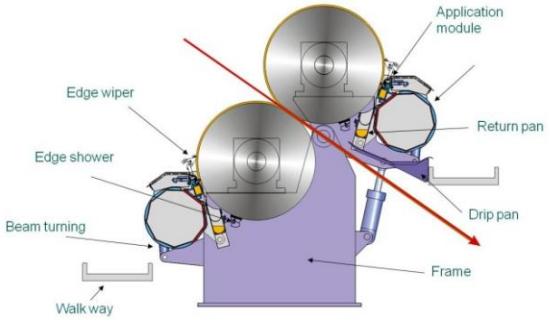
Mekaanisten käyttöjen näkökulmasta kalanterit ovat erilaisia kuin paperikoneen märän pään rakenneryhmät, koska kalentereilla ei ole kudoksia, joiden välityksellä teho- ja voitaisiin siirtää teloilta toisille. Teho siirtyykin yksinomaan nippien ja paperiradan itsensä kautta. Katkojen minimoimiseksi ja paremman prosessin ylösajettavuuden takia pääosa kalanterin teloista on erikseen käytettyjä. Tämän takia käyttöjä on lukuisia, mutta toisaalta tehontarve jakautuu näiden välille suhteellisen tasaisesti, jolloin keskimääräinen moottorikoko on pienempi kuin märänpään käytöissä.

Moninippikalanterissa mekaanisten käyttöjen kannalta erityispiirre on telojen tiheys. Suuren telatiheyden takia ei moottoreiden sijoittelulle jää riittävästi tilaa. Tästä syystä välitelat käytetään liian pienitehoisilla moottoreilla, jotta ne saadaan mahtumaan rungolle. Sen sijaan fyysisesti suurempien SYM-telojen moottorit ylimitoitetaan niin tehokkaiksi, että ne voivat nippien kautta välittää mitoituksellisesti puuttuvan tehon myös väliteloille. [Isokääntä 2007]

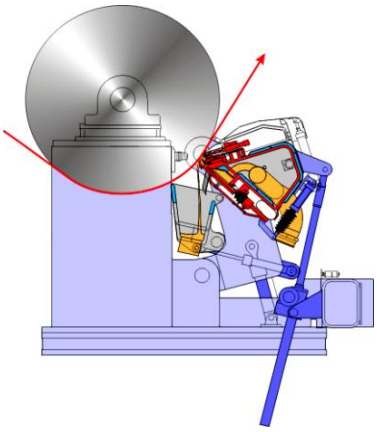
3.2.5. Päällystys- ja liimausasemat

Päällystysasemien tehtävä on joko liimata tai päällystää paperirataa. Liimauksella tarkoitetaan toimenpidettä, jossa paperirainaan syötetään täyteliimaa, joka tekee paperin rakenteesta kestävämmän ja tasoittaa pinnanlaatua. Prosessin kannalta ajateltuna liimattu paperiraina aiheuttaa vähemmän ratakatkoja, sekä pölyttämistä, koska paperi on kestävämpää ja kuidut ovat sitoutuneet adhesiivisesti liima-aineella toisiinsa. Päällystyksen tehtävänä on sen sijaan ensisijaisesti parantaa paperin pinnanlaatua. Päällystetty paperi on loppuasiakkaan näkökulmasta ”laadukkaamman” tuntuista ja näköistä. Tämän lisäksi

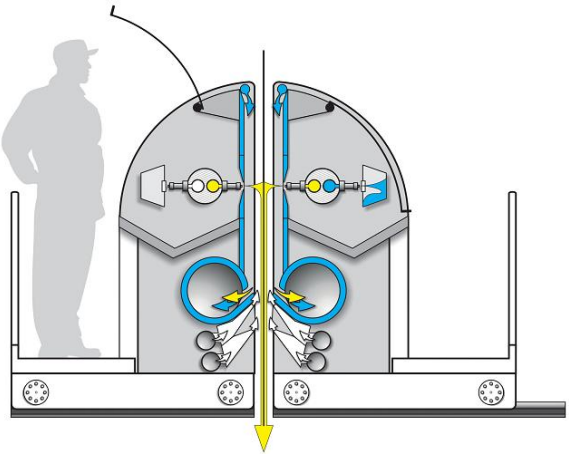
päällystetyn paperin pinta soveltuu paremmin painamiselle ja printtaukselle, minkä takia esimerkiksi pakkauskartonkien toinen puoli päällystetään, jotta niistä saadaan tehtyä mm. näyttävillä painatuksilla varustettuja myyntipakkauksia.



Kuva 3.13 *– OptiSizer [Product vault]*



Kuva 3.14 *– OptiBlade [Product vault]*

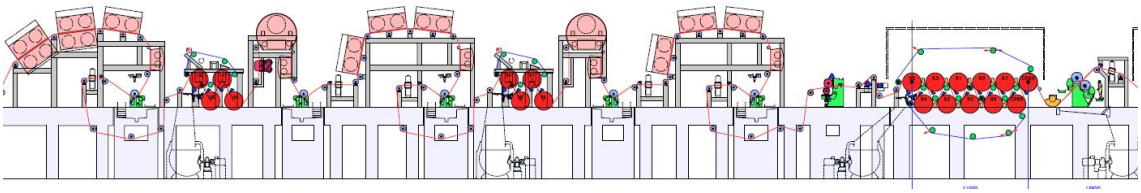


Kuva 3.15 *– OptiSpray [Product vault]*

Päällystystä ja liimausta voidaan tehdä useanlaisilla konsepteilla, jotka poikkeavat toisistaan huomattavastikin. Päällystys voidaan tehdä muun muassa:

- *Filmipäällystyksellä*, jossa päällystysaine applikoidaan ensin teloihin, joilta päällystysaine siirretään nipin välityksellä rataan. (*kuva 3.13*)
- *Teräpäällystyksellä*, jossa päällystysaine applikoidaan suoraan paperiin päällystysterän avulla. (*kuva 3.14*)
- *Suihkupäällystyksellä*, jossa päällystysaine suihkutetaan suoraan paperirainalle ilman mekaanista kontaktia (*kuva 3.15*)

Päällystysasemilla käyttöjen kannalta huomattavaa on se, että positiota saatetaan ajaa tietyissä tilanteissa nippi auki, jolloin käyttötehoa ei voida siirtää nipin yli. Tästä syystä kumpikin päällystysaseman teloista on erikseen käytettyjä. Toisaalta päällystysalueen muille teloille pätee sama kuin kalanterilla – kudoksia ei ole, joten myös muut telat tarvitsevat omat käytöt.

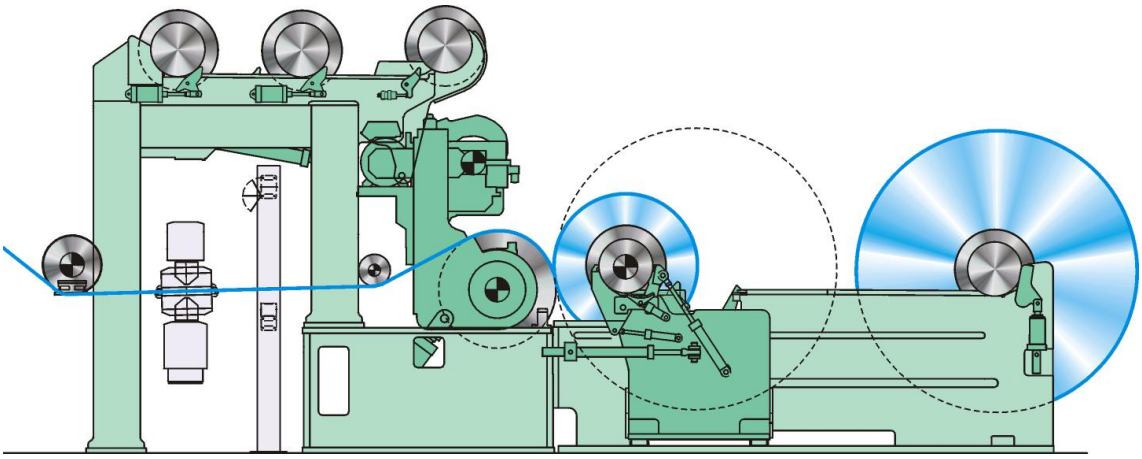


***Kuva 3.16** – Esimerkki pitkästä päällystysosasta [Metso's internal reference drawings]*

Päällystysasemalla laahaavia elementtejä on suhteellisen vähän ja nippivoimat ovat pieniä, joten käytöiltä ei vaadita suuria tehoja suhteessa muuhun paperikoneeseen. Toisaalta päällystys tuo rataan kosteutta, joten päällystyksen jälkeen paperirata tarvitsee kuivatusta ja joskus prosessilta vaaditaan erilaisia päällystemääriä eri puolille paperirataa, jonka takia päällystysosuus paperikoneesta voi sisältää paljon radan "kääntöjä" ja tilan säästämiseksi käärmemäistä mutkittelua. Tämän takia päällystysosalla voi olla huomattavia määriä teloja. Tämä nostaa päällystysosuuden mekaanistenkäyttöjen kokonaiskustannuksia. *Kuvassa 3.16* on esimerkki Metson toimittaman kartonkikoneen päällystysosasta, joka sisältää noin sata käytettyä telaa. [Isokääntä 2007]

3.2.6. Rullain

Rullaimen (*kuva 3.17*) tehtävänä on tehdä konerullia. Rullain voi olla joko loppu- tai välirullain. Loppurullaimella tarkoitetaan rullainta, joka tekee konerullia, jotka sisältävät lopullista, mutta leikkaamatonta paperia tai kartonkia. Välirullaimella tarkoitetaan rullainta, joka rullaa konerullia, jotka sisältävät muuta kuin lopputuotetta. Esimerkiksi kuivaosan jälkeen voi olla välirullain, josta konerullat toimitetaan eri jatkokäsittelylinjoille riippuen siitä, mitä paperille halutaan tehdä. Välirullaus voidaan tehdä esimerkiksi ennen kalanterointia tai päällystystä tai näiden yhdistelmää.



***Kuva 3.17** – OptiReel-rullain [Metso Prodcut Vault 2011]*

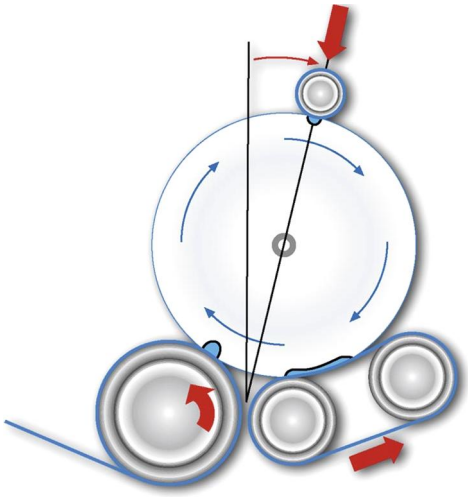
Rullaimella tärkeitä ominaisuuksia ovat sopiva kireys ja sujuvat rullanvaihdot, jotka mahdollistavat hyvin ajettavan prosessin, sekä sopivan konerullan tiukkuuden. Rullaimella käytöt sijaitsevat yleensä paperinjohtotelalla. Rullaussylinteriä ja tampuurirautaa pyörittävät ensiö- ja toisiokäytöt. Tampuuriraudat eivät ole staattisia teloja, vaan telat kulkevat prosessissa fyysisesti eteenpäin. Tämän takia tampuurirautaa pyörittävät moottorit ovat keskiökäyttöjä, jotka vaativat erikoiskytkimen, joilla ne liitetään tampuuriin. Koska tampuurin keskiökäytöt liikkuvat prosessin kuluessa, ei keskiökäyttöjen kanssa voida käyttää kiertovoitelua, koska voiteluputkistoa ei voida rakentaa tällaiseen kohteeseen luotettavasti. Tästä syystä keskiökäyttöjen kanssa käytetään roiskevoitelua, joka on muutoin paperikoneen vaihteissa epätyypillinen vaihtoehto. [Isokääntä 2007]

3.2.7. Pituusleikkuri

Pituusleikkuri on aina ’off-line’-laite, eikä näin ollen sinänsä kuulu paperikoneeseen, vaikka se löytyykin usein osana paperinvalmistuslinjaa. Pituusleikkauksen tehtävä on asiakasrullien muodostaminen. Itse paperin leikkaaminen tapahtuu kahden pyörivän terän avulla. Toinen teristä on primääriterä, jolla on pieni käyttö ja toinen on käytötön sekundaariterä, joka pyörii ensimmäisen terän avulla. Leikkausteräpisteiden määrä riippuu suoraan asiakasrullien ja koneleveyden suhteesta. Asiakasrullien leveyden muodostavien terien lisäksi pituusleikkurissa on lisäksi reunaleikkurit, jotka poistavat radasta heikkolaatuiset reunat. [Turkki 2005]

Erilaisten pituusleikkurikonseptien suurimmat erot löytyvät rullaustavasta. Perinteisin tapa on yhden tai kahden kantotelan päällä tapahtuva rullaus (*kuva 3.18*). Tässä metodissa haasteita prosessille aiheuttaa suureksi kasvavat nippikuormat, kun koko asiakasrullan paino kohdistuu melko kapeille alueille. Tämä aiheuttaa erilaisia ongelmia, kuten kreppirynkkyä, repeämiä, rullautumisen epätasaisuutta, vanaisuutta yms.

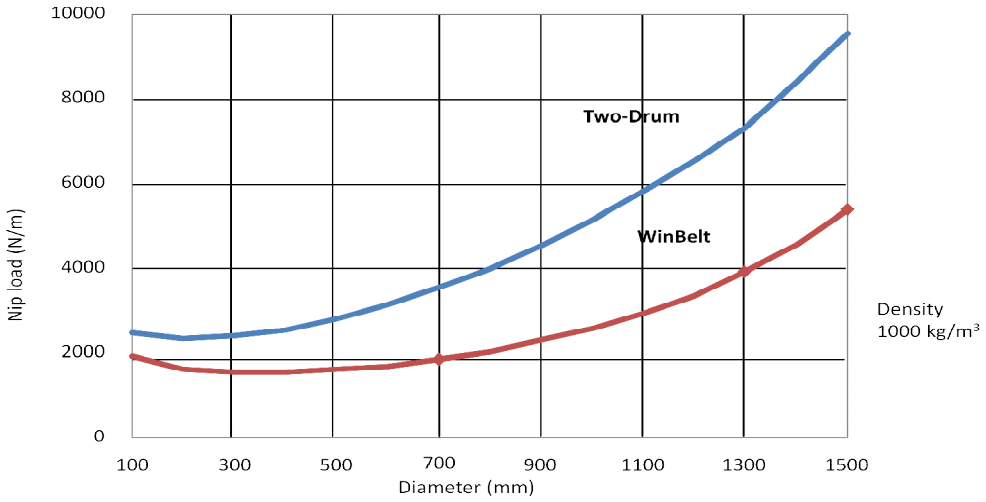
***Kuva 3.18** – WinDrum [Product vault]*



***Kuva 3.19** – WinBelt [Product vault]*

Näitä ongelmia ratkaisemaan kehitettiin Metson WinBeltin (*kuva 3.19*) tyyppiset leikkurin rullaimet, joissa toinen kantoteloista korvataan kahden telan väliin kiristetyllä hihnalla. Tämä pienentää nippipainetta (*kuva 3.20*), kun rullan paino jakautuu tasaisesti

konesuunnassa pidemmälle alueelle. Tämä mahdollistaa paremman ajettavuuden ja korkeammat tuotantonopeudet.



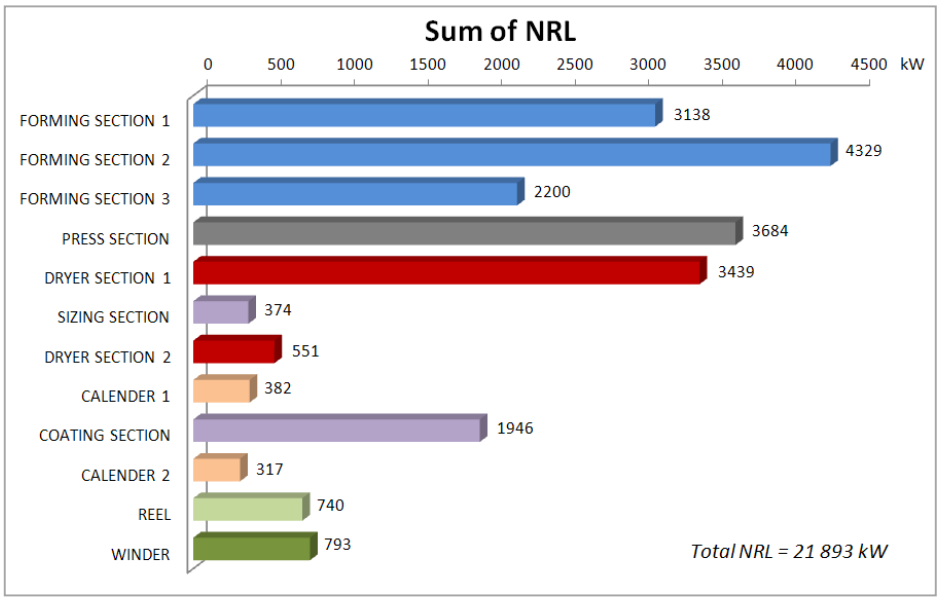
***Kuva 3.20** – WinBelt vs Two Drum - nippikuormatarkastelu [Product vault]*

Linjakäyttöjen kannalta katsottuna suuritehoisimmat moottorit löytyvät rullanmuodostuksesta, sekä pituusleikkurin vaatimasta aukirullauksesta, jossa kymmenien tonnien painoinen konerulla pitää saattaa kehänopeuteen, joka on jopa yli 2000 m/min.

3.3. Linjatason käyttötehojen esimerkkitarkastelu

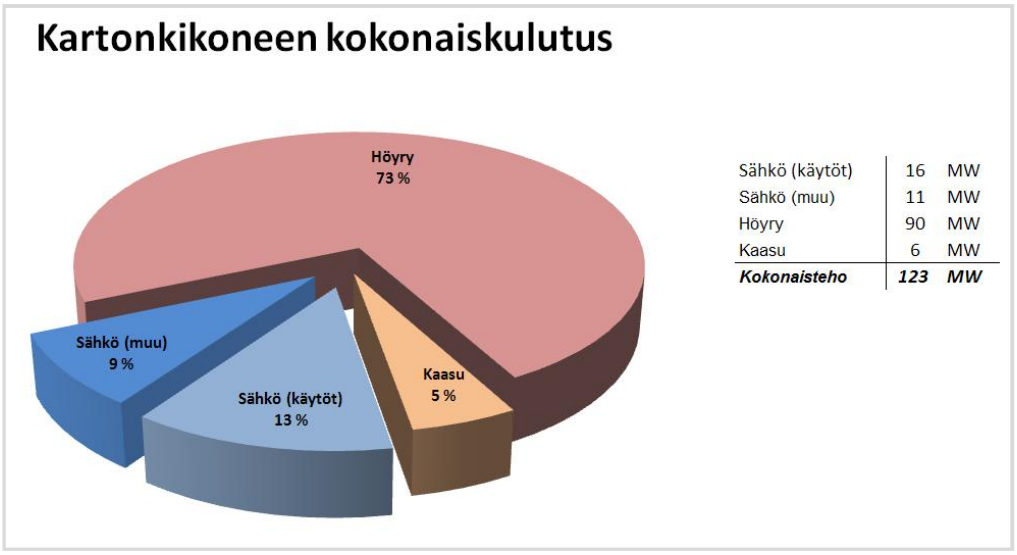
Linjatasolla eri toimilaiteryhmien väliset tehojen ja käyttöpistemäärien väliset erot ovat huomattavia. Kokonaiskuvan selkiyttämiseksi *kuvissa 3.21-3.23* on tarkasteltu mekaanisten käyttötehojen jakautumista linjan yli tehollisesti, käyttöpistemäärällisesti ja keskimääräisen moottorikoon mukaan. Tarkastelu on tehty eräästä suuresta kartonkikoneprojektista, joka käsittää prosessijärjestyksessä seuraavat rakenneryhmät:

1. Monitasoviiraosa (Forming section 1-3)
2. Puristinosa (Press section)
3. Etukuivaosa (Dryer section 1)
4. Liima-asema (Sizing section)
5. Jälkikuivaosa (Dryer section 2)
6. Välikalanteri (Calender 1)
7. Päällystysosa (useita päällystysasemia ja ilma-/infrakuivaimia) (Coating section 2)
8. Loppukalanteri (Calender 2)
9. Rullain (Reel)
10. Pituusleikkuri (Winder)



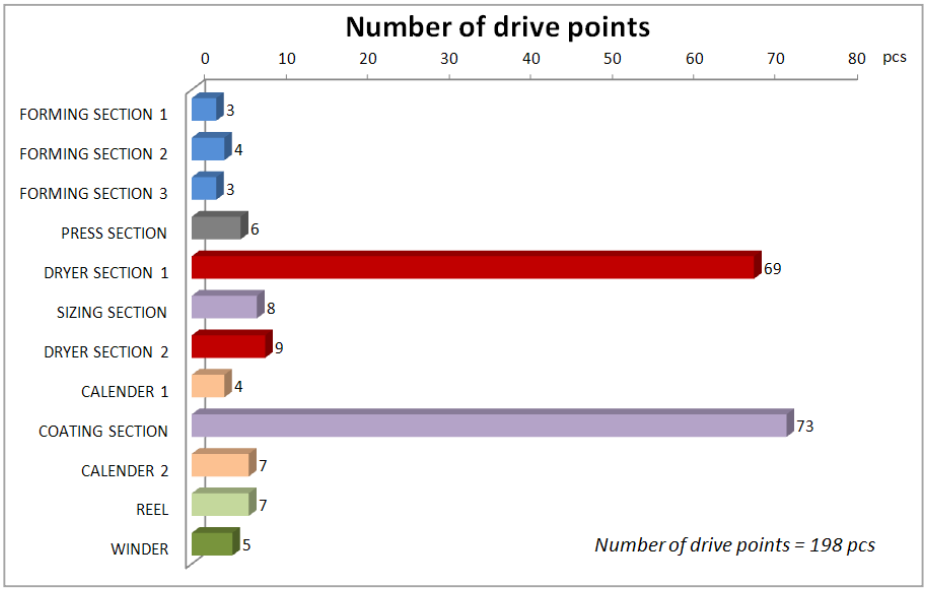
***Kuva 3.21** – Erään suuren kartonkikoneprojektin linjatason NRL-teho*

Kuvasta 3.21 voidaan huomata, että monitasoviiralla toteutetussa kartonkikoneessa viiraosa on suurin käyttötehoa kuluttava rakenneryhmä. Sen osuus koko kartonkikoneen käyttötehoista on noin 44%. Tämän jälkeen suurimmat tehon kuluttajat ovat melko tasaisesti puristin- ja etukuivaosa (~16% linjan kokonaiskäyttötehosta). Kuivan pään ryhmistä suurimmaksi tehonkuluttajaksi osoittautuu tässä kartonkikonelinjassa päällystysosa. Täytyy kuitenkin huomioida, että tarkastellun kartonkikonelinjan päällystysosa on pitkä ja monimutkainen – se sisältää useita päällystysasemia, sekä kuivaimia. Kuivan pään rakenneryhmät syövät yhteensä linjan linjakäyttöjen kokonaistehosta reilu 23 prosenttia. Koko linjan tasolla voi installoitua olla tehoa lähes 22 megawattia (normaalissa ajotilanteessa kulutus on kuitenkin lähempänä 10 megawattia), joka vastaa suurta osaa koko kartonkikoneen sähkötehonkulutuksesta, joka tässä konseptissa on noin 27 megawattia.



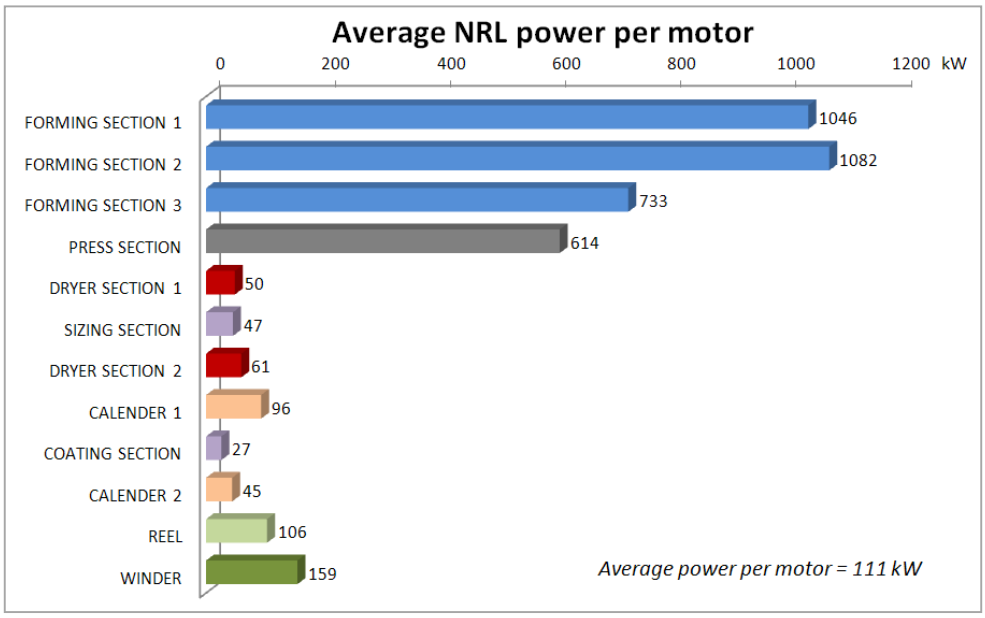
***Kuva 3.22** – Erään suuren kartonkikonelinjan arvioitu kokonaiskulutus*

Kuvasta 3.22 voidaan nähdä kuinka linjakäyttöjen installoitu teho suhteutuu koko kartonkikonelinjan kulutukseen. Taulukossa höyryn ja kaasun kulutukset on muunnettu samaan energiayksikköön, jotta lukuja voidaan vertailla. Tästä nähdään, että veden haihduttamiseen käytettävä höyry vastaa suurimmasta tehokaivosta (73% kokonaistehosta). Tämän jälkeen seuraavaksi suurin tehonkuluttaja ovat linjakäytöt (13% kokonaistehosta), jotka vastaavat selvästi suurimmasta roolista sähkönkulutuksessa. Jos kaikki tehokaivot muunnetaan sähköekvivalenttiin, niin voidaan huomata, että yksi suuri paperikone kuluttaa pienen voimalaitoksen verran energiaa, joka selittääkin miksi paperitehtaiden yhteyteen perustetaan usein myös voimalaitos. Suuren energiatarpeen lisäksi vahva selittäjä voimalaitoksen sijoitukseen paperikoneen yhteyteen on myös paperinvalmistusprosessin höyryntarve, joka pystytään tehokkaasti tyydyttämään voimalaitoksen energiantuotannon sivutuotteella, joka on sähköntuotannosta hyödyntämättä jäävä matalapainehöyry.



***Kuva 3.23** – Linjakäyttöjen käyttöpisteiden lukumäärät rakenneryhmittäin tarkasteltuna (esimerkki suuresta kartonkikoneesta)*

Käyttöpisteiden määrien suhteen tilanne on taas täysin päinvastainen kuin tehoissa. *Kuvasta 3.23* nähdään, että selvästi eniten käyttöpisteitä löytyy päällystysosalta, sekä etukuivaimelta. Tämä selittyy etukuivaimen osalta suuren telamäärän perusteella (lähes 350 telaa tai sylinteriä), vaikka tällä osuudella voimaa voidaankin siirtää kudosten välityksellä telalta toiselle. Päällystysosuudella käyttöjen suuri määrä johtuu taas (kuten kappaleessa 2.2.6 todettiin) siitä, että lähes kaikki telat joudutaan käyttämään, koska päällystysosalla ei ole kudoksia joita voitaisiin hyödyntää voimansiirrossa. Käyttöpisteiden lukumäärää laskettaessa etukuivaosa ja päällystysasema vastaavat 72 prosentista kaikista tämän paperikonelinjan käytetyistä teloista.



***Kuva 3.24** – Linjakäyttöjen normaaliajotilanteen keskimääräinen tehontarve per moottori rakenneryhmittäin tarkasteltuna (esimerkki suuresta kartonkikoneesta)*

Edellisten kuvaajien perusteella voidaan tehdä laskelma, jonka perusteella voidaan arvioida kunkin rakenneryhmän keskimääräinen käyttöpistekohtainen tehontarve (*kuva 2.24*). Tästä huomataan, että suurimmat moottorit sijaitsevat märänpään alueella: viira- ja puristinosilla, missä yksittäisten käyttöpisteiden keskimääräinen teho on useita satoja tai jopa yli tuhat kilowattia.

4. TEORIA - SYSTEEMIEN KEHITYS

Työn perusteella on tarkoitus rakentaa järjestelmä, joka tukee ja sujuvoittaa mitoitusprosessia kokonaisvaltaisesti. Tämän takia on tärkeä käsitellä asioita, joiden pohjalta toiveisiin soveltuvaa järjestelmää voidaan lähteä kehittämään. Toteutettavalle järjestelmälle asetettuja toiveita on muun muassa seuraavat:

- Helposti laajennettava karkealla tasolla (rakenneryhmät, kokonaistulokset)
- Helposti laajennettava/päivitettävä yksityiskohtaisemmalla tasolla (laskentakaavat, uudet laitteet)
- Ylläpidon on oltava hajautettavissa
- Systeemin on tuettava helppokäyttöisyyttä

Tässä kappaleessa käydään läpi systeemien kehityksen teoriaa, jonka pohjalta mekaanisten käyttöjen mitoituksen linjatason integraatiota tukevaa järjestelmää lähdetään rakentamaan. Tämä teoriaosuus on jaettu seuraaviin kokonaisuuksiin:

- Systeemien modulaarisuus
- UML
- Systeemien kehitysprosessit
- Clean wheel-metodi
- Konfigurointi

4.1. Systeemien modulaarisuus

Jotta modulaarisuudesta voidaan puhua mielekkäästi, on syytä tarkentaa käsitteet. *Komponentilla* tarkoitetaan yksittäistä osaa, joka on osa suurempaa kokonaisuutta. Esimerkiksi oven kahva voidaan katsoa komponentiksi. *Lohkolla* tarkoitetaan mitä tahansa tuotteen tai systeemin osaa. *Moduuli* on puolestaan lohko, jolla on määritelty rajapinta, jonka kautta se liittyy moduulijärjestelmään. *Moduulijärjestelmä* on yksinkertaisuudessaan lohkorakenne, joka sisältää lohkojen/moduulien vaihtokelpoisuutta eli järjestelmä on mukautettavissa tiettyjen rajoitusten mukaisesti kulloiseenkin tarpeeseen. [Ro, Fixon & Liker 2006]

Usein modulaarisuus käsitteenä liitetään fyysisiin tuoteisiin, mutta kuten Ro, Fixon & Liker esittävät artikkelissaan ”Modularity and supplier involvement in product development” (2006) voidaan modulaarisuuden avulla käsitellä myös abstrakteja systeemejä ja jopa oganisaatioita.

Nykyinen mekaanisten käyttöjen tarjousprosessi on hajautettu karkeasti viiden mitoittavan tahon välille: paperikoneelle (sisältäen viira-, puristin- ja kuivatusosan),

kalentereille, päällystysasemille, rullaimille ja leikkureille. Tämä tarkoittaa sitä, että kun asiakkaalta saadaan riittävät tiedot paperikoneen rakenteesta, laskee kukin taho oman osansa käyttötehoista ja tämän jälkeen yksi henkilö manuaalisesti parsii tulokset yhteen asiakasdokumenttiin. Uusien toiveiden mukaisesti laskenta pitäisi kuitenkin pystyä suorittamaan yhdestä paikasta, joka tällä hetkellä on vaikeaa, koska kullakin taholla on omanlaisensa laskentapohjat ja ohjelmat, joilla tulokset muodostetaan. Olemassa oleviin laskentaohjelmiin sisältyy kuitenkin paljon tietoa ja tehtyä työtä, joten ei ole mielekästä heittää kaikkia vanhoja laskentapohjia ”romukoppaan” ja aloittaa puhtaalta pöydältä. Siksi onkin mielekästä tutkia voitaisiinko laskentaohjelmia modularisoida, niin että jo laskentapohjien eteen tehty työ voitaisiin hyödyntää, mutta silti saada aikaan kokonaissysteemi joka on yhteensopiva ja yhdenmukainen kaikkien rakenneryhmien kesken.

Jotta vanhoja laskentapohjia voitaisiin hyödyntää, mutta samalla saada yhteensopiva kokonaisuus modularisoinnin avulla, täytyy tarkastella mitä vaatimuksia modularisointi asettaa systeemin eri osille.

Modulaarisuudesta voidaan katsoa olevan sekä hyötyä, että haittaa. *Kuvassa 4.1* on nähtävissä ”Fundamentals of product modularity”-kirjan [Ulrich & Tung 1991] mukaisesti eriteltynä moduloinnin positiivisia ja negatiivia asioita. Kirjan näkökulma on fyysisissä tuotteissa, mutta monia esille nostettuja asioita voidaan projisoida myös abstraktimpien systeemien modulaarisuuteen, kuten esimerkiksi kyseessä olevaan mitoitusohjelman rakenteen arkkitehtuuriin.

Edut	Haitat
+ Massavalmistuksen edut komponenttituotannossa	- Kiinteä tuotearkkitehtuuri
+ Tuotteen muutoksen helpottuminen	- Suorituskyvyn optimointi
+ Tuotevariointi	- Kopioinnin helppous
+ Läpimenoaika	- Lisääntyneet muuttuvat kustannukset (=valmistuskustannukset)
+ Tehtävien erottaminen (tuotannossa, suunnittelussa)	
+ Osaamisen keskittyminen	- Liiallinen tuotteiden samankaltaistuminen
+ Komponenttien testaus ja varmistus	
+ Kulutusosien erottaminen	
+ Sovittaminen tuotantoon ja käyttöön	
+ Vianetsintä ja korjaus	

***Kuva 4.1** – Modulaarisuuden edut ja haitat [Ulrich & Tung 1991]*

Kuvan 4.1 taulukon sisällöstä voidaan poimia mitoitusohjelmiston suunnitteluun sopivia kohtia. Modulaarisuuden hyviä, hyödynnettäviä mahdollisuuksia tarjoat muun muassa seuraavat edellisessä taulukossa listatut kohdat:

- *Tuotteen muutoksen helpottuminen*. Mitoitusohjelmalta vaaditaan helppoa muokattavuutta.

- *Tuotevariointi*. Tämän ansiosta ohjelma mukautuu kulloiseenkin tarpeeseen joustavammin.
- *Tehtävien erottaminen.* Tämän ominaisuuden avulla voidaan implementoida hajautettu ylläpito suunnittelun ja myynnin välille, kun on selvää mikä moduuli kuuluu minkäkin osaston/henkilön vastuualueeseen.
- *Osaamisen keskittyminen*. Ylläpidon hajautuksen ansiosta myös ohjelman ydinosaamiset keskittyvät niille tahoille, jotka mitoitusta normaalistikin tekevät.
- *Komponenttien testaus ja varmistus.* Pienemmät moduuli-kokonaisuudet tekevät kokonaisuuden testaamisesta helpompaa. Testaamiset voidaan hoitaa yksittäisille mitoituspalikoille ja pääohjelman platformille erikseen.
- *Vianetsintä ja korjaus*. Myös mahdollisten vikojen paikannus on helpompaa modulaarisuuden ansiosta. Kun huomataan virhe jossain laskennan osassa, voidaan modulaarisuuden ansiosta vikaa helposti lähteä etsimään oikeasta suunnasta.

Modulaarisuuden huonoja puolia ohjelma-arkkitehtuurin kannalta:

- *Kiinteä tuotearkkitehtuuri.* Modulaarisuuden hyödyntäminen asettaa kokonaisarkkitehtuurille vaatimuksia ja sen takia joitain asioita joudutaan määrittämään suhteellisen tarkasti, että modulaarinen kokonaisuus voi toimia. Arkkitehtuurista voidaan kuitenkin rakentaa suhteellisen joustava, jolloin kiinteän arkkitehtuurin haitat saadaan minimoitua.
- *Suorituskyvyn optimointi* aiheuttaa haasteita, koska modulaarisuus aiheuttaa enemmän tiedonsiirtoa rajapintojen läpi, joka hidastaa kokonaistoimintaa. Lisääntynyt laskenta-ajan kulutus on kuitenkin rajallista ja mitoitusten integraatiolla saavutetut kokonaisaikahyödyt ylittävät modulaarisuuden takia kasvavan laskenta-ajan moninkertaisesti.

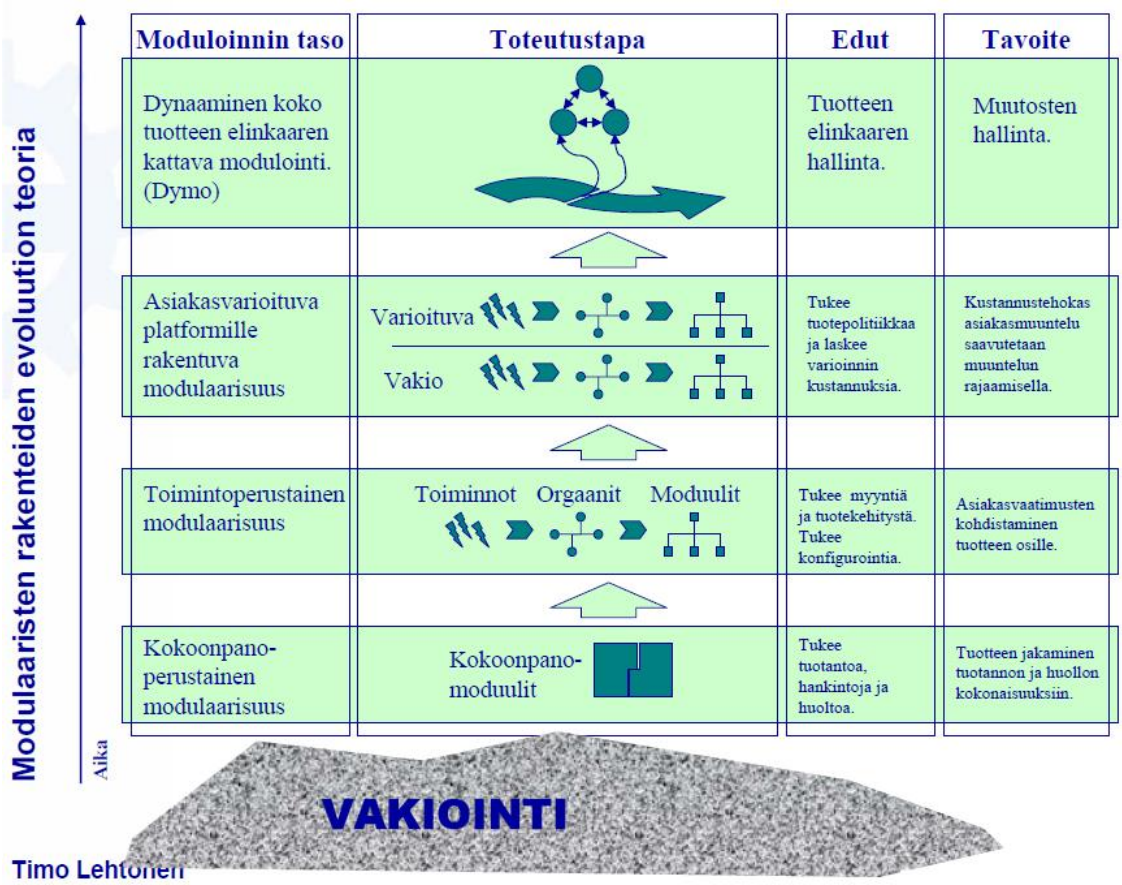
4.1.1. Modularisoinnin asteet

Modulaarisuuden tasoa voidaan tarkastella modulaaristen rakenteiden evoluution teorian kautta (*kuva 4.2*). Tämän teorian perusteella koko modularisoinnin pohjana on vakiointi, joka on toki luonnollista.

Ensimmäinen modularisoinnin taso on *kokoonpanoperusteinen modulaarisuus*. Tällä tasolla kyse on vain siitä, että pyritään modularisoimaan rakenteita siten, että ne ovat helpommin valmistettavia ja kokoonpantavia. Käytännön esimerkki tästä on esimerkiksi toisen maailmansodan aikainen saksalainen sukellusvene, jonka rakenne oli paloittain modulaarinen. Tällöin päästiin näinkin ison kokonaisuuden valmistamisessa hyötymään massavalmistuksen eduista, kun rakenne oli hajotettu tarpeeksi pieniin palasiin.

Toimintoperusteinen modulaarisuus on teorian mukaan modularisoinnin toinen taso. Tällä tasolla modulaarisuutta pyritään rakentamaan rakenteen sijasta toimintojen pohjalta ja tällöin moduulien arkkitehtuurin pohjana käytetään toiminnallisia kokonai-

suuksia. Käytännön esimerkkinä tästä modulaarisuuden lajista voidaan mainita paperikoneen rakenneryhmät. Jokaisella rakenneryhmällä on oma päätoimintonsa ja ryhmiä voidaan ketjuttaa tarpeen vaatimalla tavalla, jotta saavutetaan halutunlainen lopputuote.



***Kuva 4.2** – Modularisoinnin tasot [Lehtonen 2011]*

Kolmas taso on *asiakasvarioituva plafomille rakentuva modulaarisuus*. Se muistuttaa tietyllä tapaa seuraavassa kappaleessa esiteltävää väylämodulaarisuutta. Siinä siis määritellään tietty yhteinen rajapinta, jonka kautta kokonaisuutta voidaan yksittäisten moduulien kautta rakentaa. Tästä tasosta käytännön esimerkkinä toimii esimerkiksi pöytätietokone. Siinä tietylle platformille voi rakentaa hyvinkin erilaisia kokonaisuuksia asiakastoiveiden/-vaatimusten mukaisesti.

Neljäs ja korkein taso on *dynaaminen koko elinkaaren kattava modulointi (Dymo)*. Tämä on haastava taso ja siitä on vaikea antaa yksittäistä, nopeasti selitettävää käytännönläheistä esimerkkiä. Toisaalta tämä modulaarisuuden taso ei ole vielä lyönyt itseään läpi laajassa mittakaavassa. Tämän modulaarisuuden tason riittävä käsittely vaatisi oman laajan työnsä.

Nyt kehitettävän mitoitusohjelman modulaarisuusaste asettuu kolmannelle tasolle, eli asiakasvarioituvalle, platformille rakentuvalle modulaarisuuden tasolle. Kokoonpanoperusteinen modulaarisuuden taso on tämäntyylisessä puoliabstraktissa systeemissä

jokseenkin turha, eikä se tarjoa isoa lisäarvoa orgaanisesti rakentuneeseen epämodulaariseen kokonaisuuteen verrattuna. Toimintoperusteinen modulaarisuus on sen sijaan idealtaan oikea – kokonaisuus halutaan pilkkoa toimintakokonaisuuksien kokoisiin moduuleihin, mutta toimintoperusteinen modulaarisuus ei tarjoa yhtä joustavaa laajennettavuutta kuin systeemin vaatimuksena on. Tästä syystä määritellylle platformille rakentuva, laajennettavissa oleva modulaarisuus palvelee käyttötarkoitusta paremmin. Dymo on sen sijaan filosofialtaan liian raskas, jotta sitä voitaisiin implementoida käyttöön tämän systeemin osalta. Kokonaisuuden palaset ja prosessit ovat toistaiseksi liian erilaisia ja Dymoa tukevan rakenteen kehittäminen vaatisi liian paljon resursseja. Nyt tehtävä integraatio on kuitenkin askel yhtenäisemmän kokonaisjärjestelmän suuntaan, joka voi toimia isommassa kuvassa ensimmäisenä askeleena Dymon suuntaan, mikäli se koetaan tulevaisuudessa tarpeelliseksi.

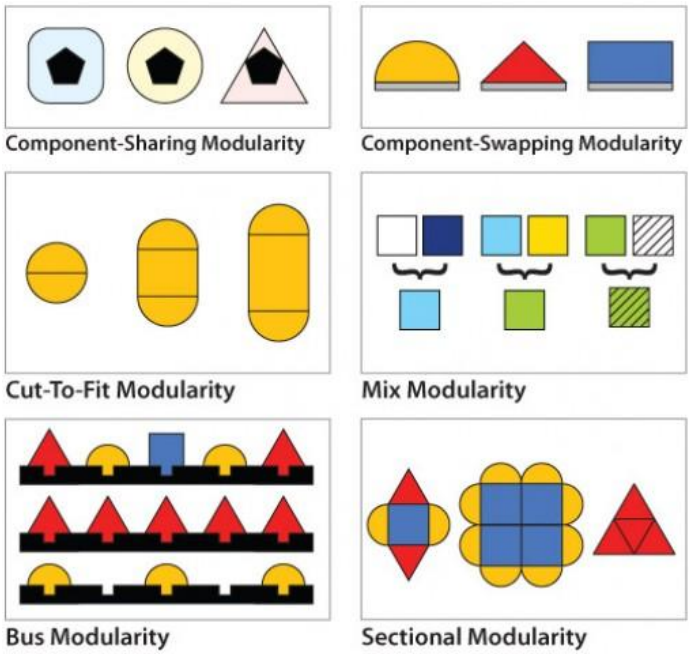
4.1.2. Modulaarisuuden tyypit

Modularisointia voidaan katsoa olevan useampaa eri tyyppiä (*kuva 4.3*). Asiaa on käsitelty jo vuonna 1978 julkaistussa artikkelissa ”Patterns of Industrial Innovation” [Abernathy 1978]. Siinä esitellään kuusi eri tapaa, jolla modulaarisuus voi ilmetä:

- Jaetun komponentin modulaarisuus
- Komponentin vaihdettavuuden modulaarisuus
- Tarpeeseen mitoituttava modulaarisuus
- Mix modulaarisuus
- Väylämäinen (bus) modulaarisuus
- Palamodulaarinen

Kullakin modulaarisuuden muodolla on omat erityispiirteensä ja ne sopivat tiettyiin käyttötarkoituksiin paremmin kuin toiset. Toisaalta ne aiheuttavat rajapintamäärittelyille hiukan erilaisia vaatimuksia.

Jaetun komponentin modulaarisuudella tarkoitetaan modulaarisuutta, jossa yhtä komponenttia voidaan käyttää useassa eri kontekstissa. Karkea esimerkki tämän tyylisestä modulaarisuudesta on esimerkiksi pesukoneen rumpu. Useassa eri pesukonemallissa voidaan käyttää samaa rumpua, vaikka muutoin pesukonemallit saattaisivat erota toisistaan huomattavastikin.



Kuva 4.3 *– Modularisoinnin kuusi tyyppiä [Abernathy 1978]*

Vaihdettavan komponentin modulaarisuus tarkoittaa sellaista modulaarisuuden muotoa, rajapinta on määritelty siten, että siihen on mahdollista käyttää erilaisia komponentteja. Tämä modulaarisuuden laji on esimerkiksi lampuissa: Samaan lampunkantaan voidaan käyttää hyvinkin erilaisia lamppuja.

Tarpeeseen mitoittuva modulaarisuudella tarkoitetaan käyttötarkoitukseen mitoittuvaa modulaarisuutta. Tämä tyyppi on käytössä esimerkiksi joissain ruokapöydissä, joissa kokonaisuus joustaa kulloiseenkin tarpeeseen siten, että ”jatkopalat” on joustavasti mitoitettu.

Mix modulaarisuudella tarkoitetaan sellaista modulaarisuutta, jossa lohkot muodostavat aina tietyn kokonaisuuden, mutta lohkot voivat olla keskenään hieman erilaisia, jolloin kokonaisuus näyttää hieman erilaiselta riippuen mitä moduuleita kokonaisuuden kokoamiseen käytetään. Esimerkiksi keittiön kaapistot voisivat olla karkea esimerkki mix modulaarisuudesta. Voit valita erilaisia ovia, vetokahvoja, ynnä muita sellaisia, mutta kokonaisuus muodostuu silti määrätynlaiseksi.

Väylämodulaarisuudella tarkoitetaan modulaarisuutta, jossa väylään määritellään rajapinta, johon saadaan joustavasti liitettyä erilaisia moduuleita ja yleensä myös määrällisesti joustavasti. Väylämodulaarisuudesta hyvä esimerkki on tietoverkko, jossa määritellyn rajapinnan kautta voidaan liittää verkkoon huomattavasti toisistaan poikkeavia laiteita. Väylämodulaarisuus muistuttaa tietyllä tapaa vaihdettavan komponentin modulaarisuutta, mutta myös eroja löytyy. Vaihdettavan komponentin modulaarisuudessa on tyypillistä se, että komponentti tehdään alun perinkin sopimaan osaksi modulaarista kokonaisuutta, eivätkä komponentit välttämättä toteuta itsenäisesti mitään erityistä tehtävää. Väylämodulaarisuudessa nämä moduulit eivät aina ole edes suunniteltu sitä varten, että ne olisivat osa modulaarista rakennetta, vaan ne voivat olla alun perin itsenäisiä entiteettejä.

Palamodulaarinen systeemi on sellainen, jossa määriteltyjen rajapintojen kautta kokonaissysteemiä voidaan laajentaa moduuleja moduulien päälle. Tyypillinen esimerkki tästä modulaarisuuden muodosta on lego®-palikat.

Näiden lisäksi voidaan katsoa, että on myös seitsemäs modulaarisuuden laji: *plus-modulaarisuus*. Sen ovat tuoneet esille Jarmo Juhola ja Kalle Välimaa kirjoituksessaan ”Tuotevarioinnista kilpailukykyä – tarjouksesta toimitukseen” [Juhola & Välimaa 1997]. Plusmodulaarisuudessa olennainen ero tulee siitä, että se keskittyy olennaisesti asiakastarpeiden täyttämiseen seuraavan periaatteen mukaisesti: ”Moduulirakennetta luotaessa tulee pyrkiä minimäärään moduuleja, joilla voidaan tyydyttää kaikki valitut asiakastarpeet.” Ideana on siis, että kokonaisuudessaan moduuliarkkitehtuuria ei tarvitse suunnitella alussa loppuun asti, vaan uudet asiakastarpeet sijoitetaan uusiin moduuleihin sillä ajatuksella, että jokaisessa moduulissa on yksi asiakastarve. Tämän ansiosta tuotteen konfigurointi on helppoa.

Nyt rakennettavan mitoitusohjelmiston moduuliarkkitehtuurilta vaaditaan ominaisuuksia, joita ei täytä yksikään edellä esitetyistä modulaarisuuden tyypeistä yksinään. Siksi onkin mielekästä rakentaa systeemi useamman modulaarisuuden tyypin yhdistelmän kautta. Seuraavassa on listattu tyypit, joita arkkitehtuurin rakennuksessa voidaan hyödyntää:

- *Plus-modulaarisuus*. Koska kokonaismitoitusohjelma muodostetaan ennestään olemassa olevien laskentapohjien perusteella, on myös käyttäjäryhmillä erilaisia toiveita tietyistä toiminnallisuuksista. Näin ollen on hyvä hyödyntää plus-modulaarisuuden filosofiaa arkkitehtuurin taivuttamiseksi käyttäjätarpeiden mukaisesti.
- *Väylämodulaarisuus*. Ohjelmasta pyritään samaan rakenteeltaan sellainen, että se mahdollistaa uuden mitoituskokonaisuuden helpon liittämisen ohjelmakokonaisuuteen. Tähän ajatusmaailmaan väylämodulaarisuus sopii erinomaisesti. Väylämodulaarisuuden ajatuksia myötäillen tullaan rakentamaan pääohjelman platform, joka mahdollistaa laskentamoduulien joustavan liittämisen tähän. Näin kokonaisrakenne muotoutuu ajan kanssa muuttuviin tarpeisiin sopivaksi.
- *Tarpeeseen mitoittuva modulaarisuus*. Yksittäisten mitoituslaskentamoduulien sisältö elää jatkuvasti uusien konseptien ja toimilaitteiden takia. Tästä syystä laskentamoduulien suunnittelussa tulee ottaa huomioon joustavuus, jotta moduuleita voidaan laajentaa ja päivittää helposti.
- *Jaetun komponentin modulaarisuus*. Jotta linjalaajuisen laskennan synergiaetuja voidaan hyödyntää tehokkaammin, on ohjelman kyettävä hyödyntämää tiettyjä kaikille laskennoille yhteisiä muuttujia. Kun nämä paketoidaan yhteen komponenttiin, jota kaikki laskentamoduulit hyödyntävät, voidaan puhua jaetun komponentin modulaarisuudesta.

4.1.3. Modulaarisen systeemin arkkitehtuurin rakentaminen

Modulaarisen systeemin arkkitehtuurin suunnitteluun ei tarvitse lähteä tyhjältä pohjalta. Referenssiksi systeemin rakentamiseen on valittu perinteinen kehityksen V-mallin pohjalta luotu 12 askeleen prosessi (*kuva 4.4*).

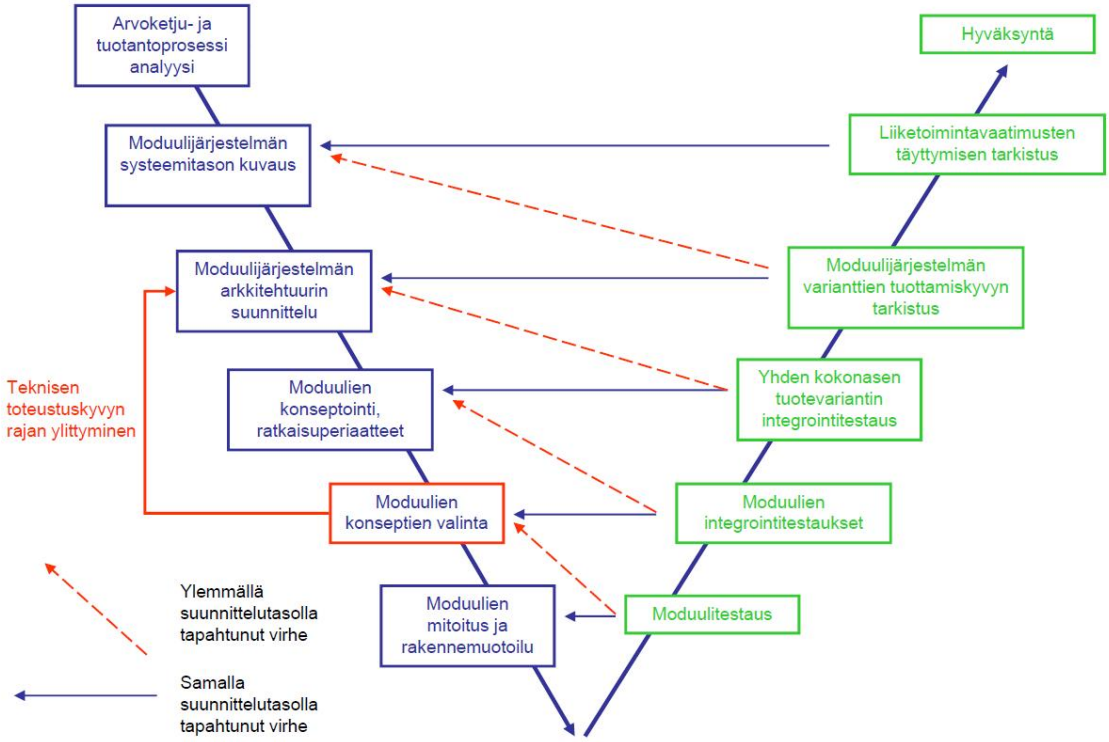
Modulaarisen systeemin rakentaminen lähtee tämän mallin mukaisesti liikkeelle *arvoketju- ja tuotantoprosessin analyysillä*. Tässä tapauksessa kyseinen analyysi on tehty kappaleessa ”2. Tarjousprosessi”. Siinä kuvataan kokonaisprosessi ja vaatimuksia joita kehitettävältä systeemiltä vaaditaan.

Moduulijärjestelmän systeemitason kuvauksen pyrkimyksenä on muodostaa moduulijärjestelmän tavoitearkkitehtuuri ja seuraavassa *arkkitehtuurin suunnittelun* vaiheessa pyritään suunnittelemaan moduulijärjestelmä, joka mukailee tavoitearkkitehtuuria ja täyttää asetetut vaatimukset. Tämä tavoitearkkitehtuurin muodostaminen ja arkkitehtuurin suunnittelun lopputulos on kuvattu kappaleessa ”2.5 Systeemin kehittäminen”.

Neljännessä askeleessa toteutetaan yksittäisten *moduuleiden konseptointi ja* listataan *ratkaisuperiaatteet*. Tämän jälkeen suoritetaan *moduulien konseptien valinta* parhaiden ehdokkaiden joukosta. Mikäli tässä vaiheessa prosessia ylitetään teknisen toteutuskyvyn rajoja, niin voidaan tarvittaessa palata kolmanteen vaiheeseen uudelleensuunnittelemaan arkkitehtuuri sellaiseksi, että se kyetään toteuttamaan. Kun konseptivalintoihin ollaan tyytyväisiä, voidaan siirtyä kuudenteen vaiheeseen, jossa *moduulit mitoitetaan ja muotoillaan*.

Tämän jälkeen siirrytään V:n toiseen haaraan eli testaus- ja tarkistusosioon. Ensimmäisenä tulee suorittaa yksittäisten *moduulien testaus*. Mikäli tässä vaiheessa löydetään virheitä, ne on yleensä tehty V-prosessin ensimmäisen haaran lopussa, jolloin virheiden korjaus ei vaadi suurta palaamista ketjussa taaksepäin. Moduulien karkea testaus suoritettiin käsiteltävän systeemin osalta osana rajapintojen, sekä moduulien rakentamista ja mahdollisia virheitä korjattiin sitä mukaa kun niihin törmättiin.

Kun yksittäiset moduulit on testattu, siirrytään prosessissa seuraavaksi *moduulien integraatiotestaukseen*. Tässä vaiheessa pyritään varmistamaan, että kokonaismoduulijärjestelmä tukee moduulien liittämistä halutulla tavalla ja yksittäiset moduuliintegroinnin toimivat. Mikäli tällä tasolla on tapahtunut jokin isompi virhe, niin se saattaa olla peruja moduulien ratkaisumallien muodostamisesta. Tämän takia virheen sattuessa joudutaan prosessissa palaamaan takaisin melko paljon, jotta moduuli saadaan toimimaan halutulla tavalla osana kokonaisarkkitehtuuria. Tätä testausta tehtiin käytännössä todella paljon osana rajapintojen käytännön rakentamista ja suuremmilta virheiltä vältyttiin.



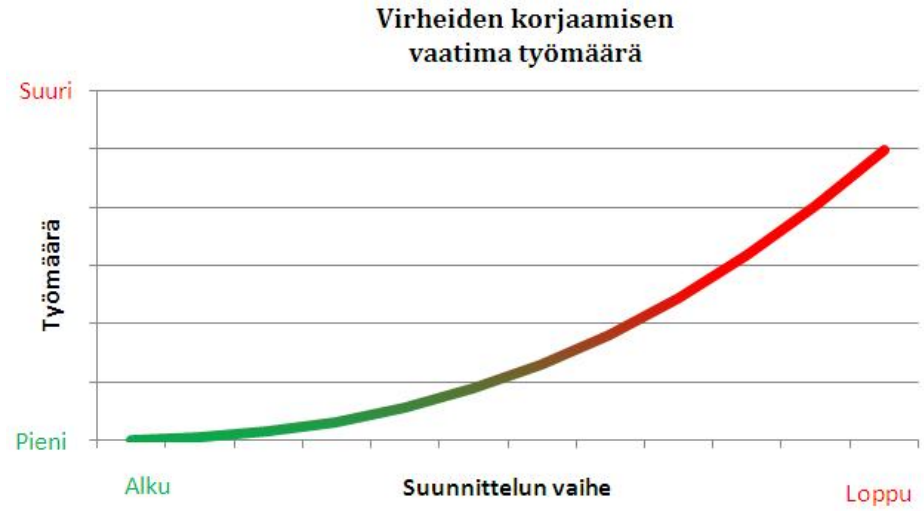
Kuva 4.4 *– Askeleet modulaarisen systeemin rakentamiseen [Lehtonen 2011]*

Tuotevariantin integrointitestauksen vaiheessa testataan, että koko moduulijärjestelmä toimii yhteen halutulla tavalla, kun useampi moduuli on käytössä. Tämä testaus ei aiheuttanut tehdyn systeemin toteutuksessa ongelmia, koska edellisten vaiheiden testaukset oli suoritettu riittävällä tarkkuudella. Mikäli tässä vaiheessa olisi törmätty suurempiin virheisiin, olisi se ollut todennäköisesti peruja jo kokonaisarkkitehtuurin muodostamisesta. Jos tämän tason virhe olisi tapahtunut, olisi sen korjaaminen vaatinut kokonaisarkkitehtuurin muuttamista ja tässä vaiheessa prosessia, se olisi ollut huomattavan työlästä.

Kun tuotevarianttien integrointitestaukset on suoritettu, tulee seuraavaksi toteuttaa laajempi *varianttien tuotteistamiskyvyn tarkistus*. Kyseessä olevassa systeemissä tämän testauksen voidaan ajatella olevan sitä, että testataan kykeneekö ohjelmisto vastaamaan erilaisten kokonaislinjakonseptien vaatimuksiin. Käytännössä siis testattiin useiden erilaisten linjakokonaisuuksien laskentaa, joissa oli erilaisia rakenneryhmiä, eri järjestyksessä ja eri määrissä. Testaukset sujuivat hyvin ja vastaan tuli lähinnä hienosäädettäviä asioita.

Kun yllä kuvatut testaukset on tehty, täytyy tarkastella täyttääkö järjestelmä sille asetetut *liiketoimintavaatimukset*. Koska vastaava systeemi on ollut käytössä jo aiemmin, on syytä olettaa, että myös uusi systeemi vastaisi sille asetettuja vaatimuksia. Tämä vaihe kehitysprosessista on kuitenkin yhä meneillään ja vaatimusten täyttymisiä päästään arvioimaan paremmin, kun järjestelmän käyttäjätestaukset on tehty ja järjestelmää päästy käyttämään ensimmäisen kerran asiakasprojekteissa täydessä laajuudessaan. Mikäli järjestelmä täyttää liiketoimintavaatimukset, niin seuraavana vaihe on hyväksyntä käyttöön osana tarjous-, myynti- ja suunnittelutyötä. Uusi järjestelmä on pyritty raken-

tamaan vanhoja yksittäisiä järjestelmiä kunnioittaen ja taustalla tapahtunut suhteellisen merkittävä arkkitehtuurimuutos on järjestelmän käyttäjälle suhteellisen näkymätön. Tästä syystä muutosvastarinta lienee suhteellisen vähäistä ja ohjelma saavuttaa käyttäjien hyväksynnän helpommin kuin täysin uuden ohjelman tapauksessa.



***Kuva 4.5** – Kumuloituvan työn virheiden korjaamisen vaatima työmäärä prosessin edetessä*

Kokonaisuudessaan moduuliarkkitehtuurin suunnitteluun pätee sama sääntö kuin moniin muihinkin kumuloituvaan työhön perustuviin prosesseihin: kaikki prosessin vaiheet tulee tehdä huolellisesti, jotta virheitä ei periydy seuraaviin vaiheisiin. Mikäli virheitä periytyy ja mitä myöhemmin virhe huomataan, sitä suuremman työn ja kustannukset se aiheuttaa (kts. *kuva 4.5*).

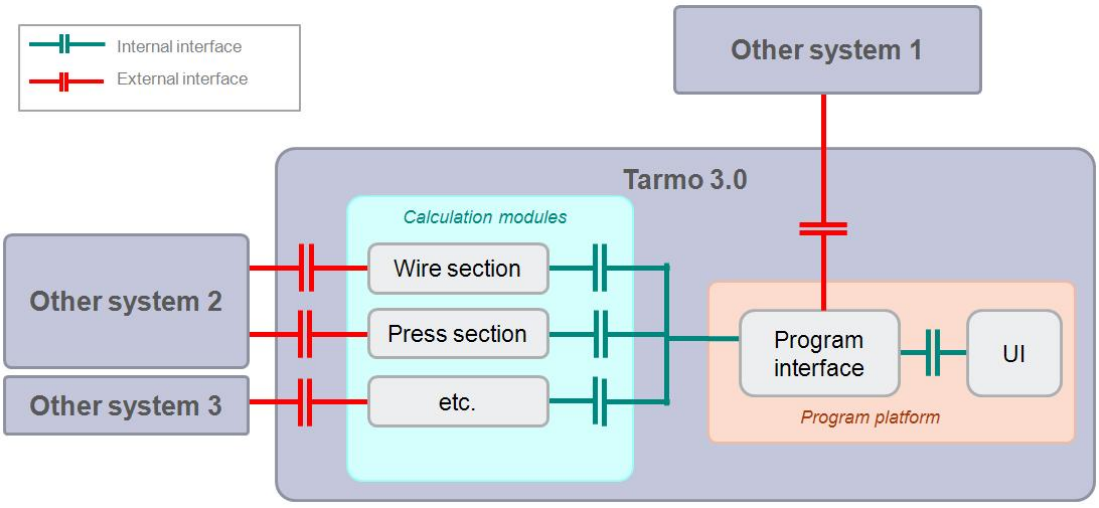
4.1.4. Moduulien riippumattomuus, rajapinnat ja resoluutio

Modulaarisen järjestelmän kehittämisessä on kaksi tärkeää asiaa, jotka on syytä pitää mielessä: riippumattomuus ja oikea koko. Jo modulaarisuuden määritelmässä tulee esille vaatimus siitä, että moduulien on oltava mahdollisimman riippumattomia toisistaan. Riippumattomuuden määrä voi vaihdella, mutta tavoitteena on oltava, että moduulien välisten rajapintojen läpi kulkisi mahdollisimman vähän ”turhia” data-, massa- tai muita virtoja. Mahdollisimman itsenäisten moduulien suunnittelussa voi tarvittaessa käyttää apuna suunnittelun riippuvuusmatriisia (Design Structure Matrix, DSM). Sen avulla pystytään klusteroimaan alkioiden riippuvuuksia ja tämän avulla voidaan miettiä moduulijakoa siten, että kukin moduuli sisältäisi mahdollisimman paljon sisäisiä ja mahdollisimman vähän ulkoisia riippuvuuksia.

Riippumattomuuteen liittyy myös rajapintojen rakentaminen. Kuo-Min Chen, Ren-Jye käsittelevät artikkelissaan ”Interface strategies in modular product innovation” (2005) kuinka tärkeässä roolissa hyvin määritellyt rajapinnat ovat modulaarisen tuotteen kehittämisessä. Mitä selkeämpiä ja standardoidumpia rajapinnat ovat, sitä joustavampia ja uusiokäytettävämpiä moduulit ovat. Tämä mahdollistaa epäsuorasti suuremman mää-

rän tuotevariaatioita. Tämä on laajemmassa mittakaavassa tärkeää, kun vastaavia moduuleita voidaan käyttää hyvin erilaisissa systeemeissä ja näin saavutetaan jo suunnitelluilla moduuleilla synergiaetuja, yhdenmukaisia toimintatapoja ja toiminnan tehokkuutta.

Sisäiset ja ulkoiset rajapintapäätökset vaikuttavat tuotteen/systeemin arkkitehtuuriin, erityisesti moduulien sijoitusstrategioihin osana modulaarista systeemiä. Rajapintastrategia voi loppupeleissä vaikuttaa jopa yrityksen strategiatasolla, kun mietitään miten avoimia rajapintoja systeemeihin rakennetaan. Tällöin on pohdittava ja linjattava millä tavoilla moduuleita tai kokonaista moduulisysteemiä jaetaan kumppaneiden tai jopa kilpailijoiden kanssa. Tästä voidaan päästä hyötymään toiminnan virtaviivaistumisena, mutta tällöin liiketoimintariskit on arvioitava tarkasti ja rajapintatoteutukset tehtävä turvallisiksi, omaa ydinliiketoimintaa suojellen. Nyt suunniteltavassa järjestelmässä sisäiset rajapinnat on määritelty niin, että tarvittaessa moduuleita voidaan käyttää myös muista systeemeistä. Ulkoista, koko systeemin kattavaa rajapintaa ei lähtökohtaisesti vaadittu, mutta se on syntynyt osana systeemin kehitystä ikään kuin sivutuotteena. Jos tulevaisuudessa tulee tarve käyttää Tarmoa esimerkiksi osana koko linjan kattavaa tehonkulutuslaskentaa, niin se on mahdollista suhteellisen pienellä vaivalla. *Kuvassa 4.6* on nähtävillä Tarmon rajapintastrategian päälinjaukset suhteessa yksinkertaistettuun moduulikaavioon. Kuvassa on nähtävissä sisäiset rajapinnat vihreällä merkinnällä ja ulkopuoliset rajapinnat punaisella merkinnällä. Kuten kuvasta voi huomata, yksitäisten laskentamoduulien käyttöön on kaksi vaihtoehtoa: moduuleita voi käyttää suoraan saman rajapinnan kautta kuin Tarmon API käyttää tai vaihtoehtoisesti voi niitä voi käyttää välillisesti Tarmon oman API:n kautta.

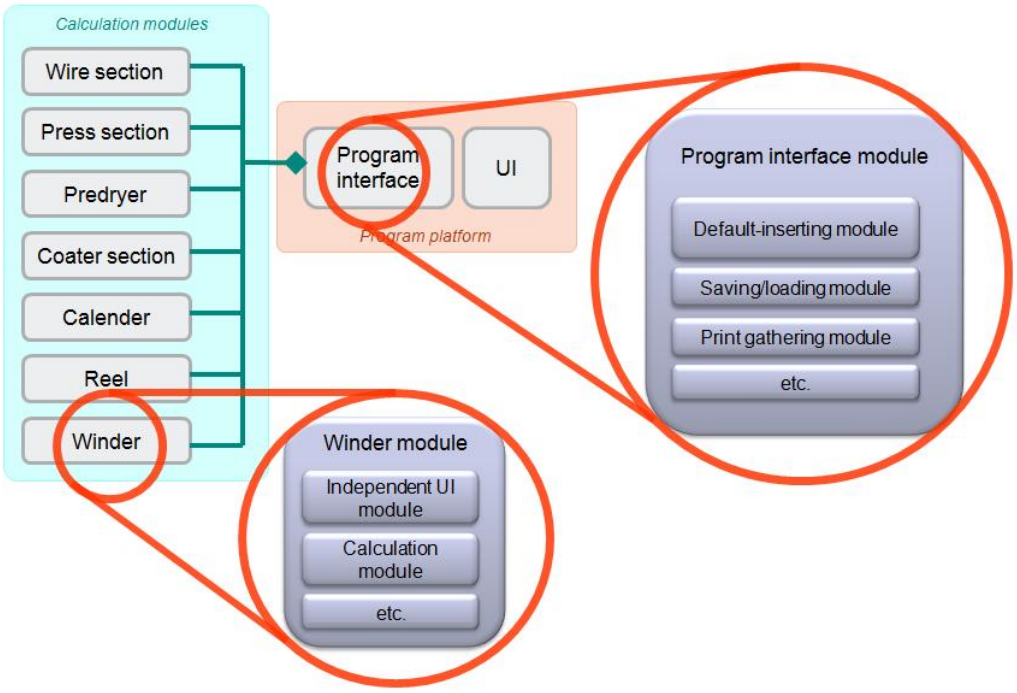


***Kuva 4.6** – Tarmo 3.0:n rajapintakuvaaja*

Luonnollisesti on tärkeää että moduulit ovat kooltaan (resoluutioltaan) käyttötarkoitukseensa sopivia. Liian suuret moduulit eivät mahdollista hyötyjä, joita moduloinnilla haetaan, koska moduulit muistuttavat liiaksi kokonaisia tuotteita. Koska moduloinnilta haetaan usein asiakasvarioituvuutta, niin variaatioita tehdessä suuria ja kalliita

moduuleita tarvitaan isoja määriä, jotta halutut toimintakokonaisuudet saadaan toteutettua. Tämä syö pääomia ja on hankala ylläpitää tuotehallinnan kannalta. Toisaalta liian pienet moduulit alkavat helposti muistuttaa normaaleja ”ämpäritavara”-komponentteja ja kokonaisuuden kasaaminen muistuttaa pitkälti moduloimattoman systeemin rakentamista.

Nyt kehitettävässä systeemissä on kiinnitetty huomiota edellä mainittuihin asioihin ja kokonaisuudesta on saatu rakennettua moduulinen rakenne, joka palvelee tarkoituksenmukaisesti järjestelmän sidosryhmiä. Laskentamoduulit on onnistuttu rakentamaan niin riippumattomiksi, että niitä pystytään tarvittaessa käyttämään yksin ilman pääohjelman platformia. Tämä tosin rajoittaa platformiin rakennettujen toiminnallisuuksien ja käyttöä helpottavien toimintojen hyödyntämistä, mutta selvä moduulien riippumattomuus on olemassa. Toisaalta pääohjelmassa käytettäviä alimoduuleita käytetään nykyään myös muissa suunnittelutyökaluissa, joka kertoo myös siitä, että myös alimoduulit ovat tietyllä tapaa riippumattomia.

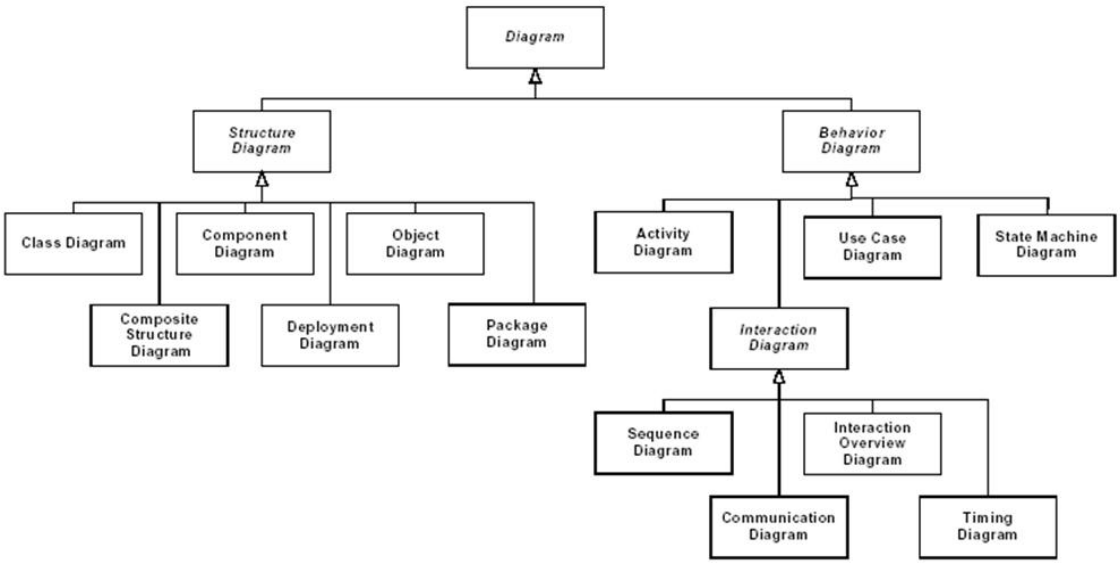


***Kuva 4.7** – Kehitettävän systeemin moduuliresoluutio: päämoduulit ja alimoduulit*

Moduulien koko rakennetussa systeemissä jakautuu kahteen kokoluokkaan: suuriin päämoduuleihin ja päämoduulien alimoduuleihin (kts. *kuva 4.7*). Päämoduulitason karkea kuvaus löytyy *kuvasta 2.7* kappaleessa, jossa käsiteltiin systeemin kokonaistason toimintaa ja sen kehitystä. *Kuvassa 4.7* on nähtävissä kuinka päämoduulit sisältävät toiminnallisia alimoduuleita, jotka viime kädessä vastaavat kokonaisuuden käytännön toiminnasta. Tarkemman moduuliarkkitehtuurin tarkastelu löytyy kappaleesta 6.1.1.

4.2. UML

UML eli unified modeling language on 1997 kehitetty standardoitu graafinen mallinnuskieli, joka kehitettiin alun perin järjestelmä- ja ohjelmistokehitystä varten. UML koostuu suuresta määrästä erilaisia kaavioita, joita voidaan käyttää erilaisten asioiden kuvaamiseen. *Kuvassa 4.8* on nähtävillä kaikki standardin mukaiset UML-kaaviotyypit ja kuinka ne jaotellaan kolmeen pääkategoriaan: rakennekaavioihin, käyttäytymiskaavioihin ja vuorovaikutuskaavioihin, joka on itse asiassa osa käyttäytymiskaavioita. [OMG 2005]



***Kuva 4.8** – UML-kaaviotypit*

- Käyttäytymiskaavio (Behavior diagram)
  - Aktiviteettikaavio (Activitity diagram)
  - Käyttötapauskaavio (Use case diagram)
  - Tilakaavio (State (machine) diagram)
- Vuorovaikutuskaavio (Interaction diagram)
  - Ajoituskaavio (Timing diagram)
  - Kokoava vuorovaikutuskaavio (Interaction overview diagram)
  - Kommunikointikaavio (Communication diagram)
  - Sekvenssikaavio (Sequence diagram)
- Rakennekaavio (Structure diagram)
  - Komponenttikaavio (Component diagram)
  - Koostekaavio (Composite structure diagram)
  - Luokkakaavio (Class diagram)
  - Oliokaavio (Object diagram)
  - Pakkauskaavio (Package diagram)
  - Sijoittelukaavio (Deployment diagram)

Francesco Basile, Pasquale Chiacchio ja Domenico Del Grosso totevat artikkelissaan “A two-stage modeling architecture for distributed control of real-time industrial systems: Application of UML and Petri Net”, että UML on tehokas työkalu systeemin vaatimusten formaaliin kuvaamiseen, sekä kontrolloidun, että kontrolloimattoman systeemin osalta. Tämän mahdollistaa UML:n karkean tason käsittely, joka yksinkertaistaa alimman tason aktoreiden toimintaa, jotta kokonaiskuvasta voidaan muodostaa helposti paremmin ymmärrettävä kokonaisuus. Tämän lisäksi UML:n mahdollistaa havainnollistamaan systeemien osien vaatimuksia, toimintoja, tehtäviä, sekä rajapintoja.

Sen lisäksi, että UML mallien käyttö helpottaa systeemin rakenteen ja arkkitehtuurin suunnittelua, se helpottaa myös systeemin ylläpitoa. Ariadi Nugroho toteaakin artikkelissaan ”Level of detail in UML models and its impact on model comprehension: A controlled experiment”, että tulokset UML:n hyödyistä ovat kiistattomat. Tarkastelun perusteella voidaan todeta, että UML:n käyttö ylläpidossa näkyy ajankäytön vähentymisenä, sekä tehtyjen korjausten parempana oikeellisuutena. Ylläpidon ajankäytöllistä hyötyä syö UML-kaavioiden päivittämisen vaatima työ tehtyjen korjausten mukaisesti. Tästä huolimatta kokonaisetu UML:n käytön hyväksi on silti selvä.

Ariadi Nughoron artikkelissa korostetaan myös sitä, että kun UML-kaavioita tehdään, täytyy olla alun perin myös selvää, että mihin kaavioita halutaan käyttää. Jos kaavioita halutaan käyttää arkkitehtuuritason suunnittelun tukena, tulee kaavioiden suunnittelutason olla karkea, jotta kaavioiden kompleksisuus ei tekisi niistä työskentelyä hidastavia elementtejä. Toisaalta jos UML-kaaviota halutaan käyttää jonkun tietyn osa-alueen tarkemman käytännöntason toteutuksen tukena, niin suunnittelutason täytyy olla yksityiskohtainen, jotta siitä voidaan aidosti hyötyä systeemin yksityiskohtaisemman tason rakentamisessa. [Nugroho 2009] Nyt tehtävissä tarkasteluissa on keskitytty ensisijaisesti arkkitehtuuritason UML-kaavioihin, mutta joistain yksittäistapauksista on tehty yksityiskohtaisemman suunnittelutason kuvaajia.

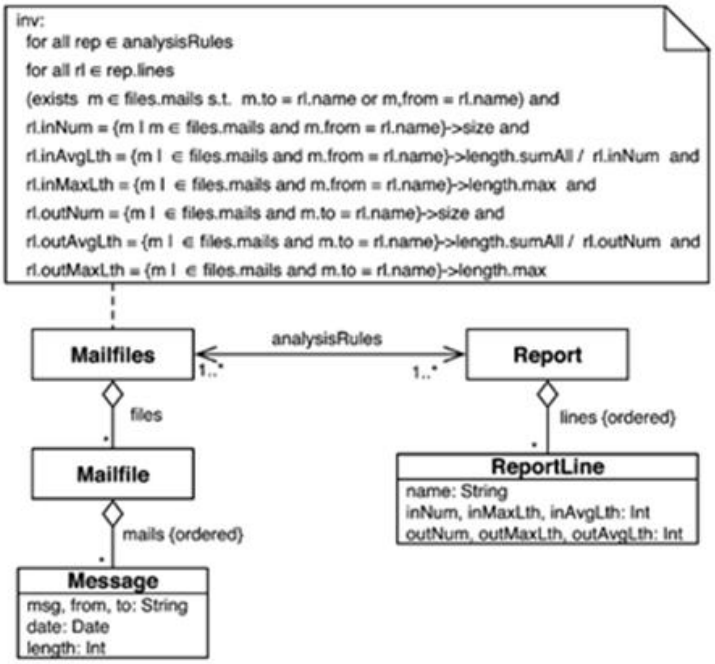
4.2.1. UML:ään perustuva kehitysprosessi [Immonen 2010]

Choppy ja Reggio esittelevät artikkelissaan [Choppy, Reggio 2005] UML:ään pohjautuvan ohjelmistokehityksen perusteita. Heidän mielestään UML:n valinnalle osaksi kehitysprosessia on useita hyviä syitä: menetelmä on laajalti tunnettu ja käytännössä testattu toimivaksi. Valmiita työkaluja UML-kaavioiden tekemiseen on lukuisia, joka helpottaa käyttöönottoa ja näin madaltaa kynnystä jalkauttaa menetelmä osaksi yritysten prosesseja. Toisaalta UML taipuu yhtä hyvin niin isojen kokonaisuuksien kuin myös pienien yksityiskohtienkin suunnitteluun, jolloin samoja menetelmiä voidaan käyttää kehitysprosessissa aina karkeasuunnittelusta hienosuunnitteluun. Haastavuudeksi Choppy ja Reggio sen sijaan kokivat sen, että UML sisältää niin suuren määrän erilaisia rakenteita ja työkaluja, että sopivien valinta voi joskus olla runsauden pulan takia haastavaa.

Choppyn ja Reggion lähestymistapa korostaa erityisesti kehitettävän ohjelmiston toiminnallisuutta ja rajapintoja, mutta myös käyttäjän asemaa osana ohjelman toimintaa.

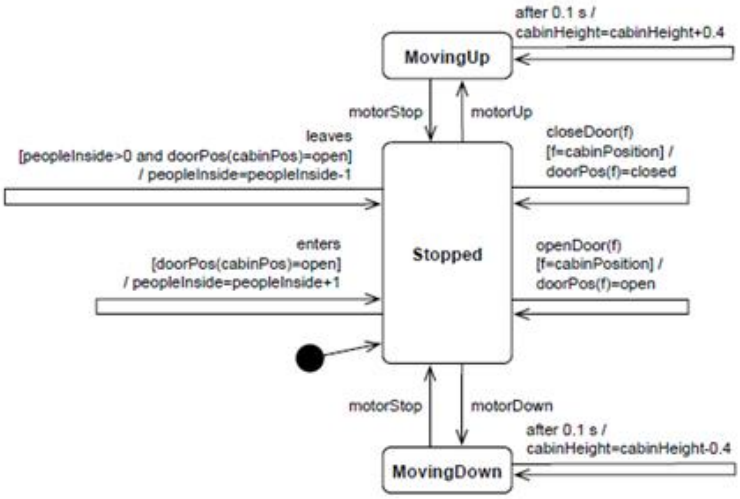
Artikkelissa esitetty UML-pohjainen kehitysprosessi sisälsi karkealla tasolla kolme vaihetta:

1. Vaikutusmallin rakentaminen (Domain Model)
2. Vaatimusten määrittely (Requirement Specification)
3. Suunnittelun määrittely (Design Specification)



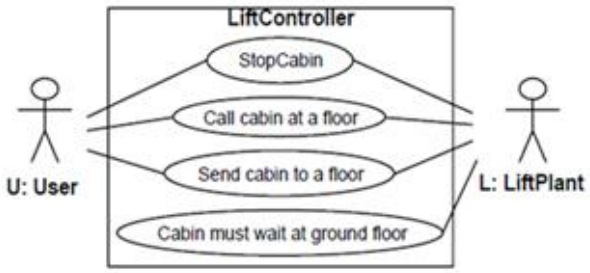
Kuva 4.9 *– Luokkakaavio*

Vaikutusmallin rakentamisessa tulee ottaa huomioon vain asiat, joita vaaditaan halutun ratkaisun muodostamisessa. Tähän Choppy ja Reggio suosittelevat UML-rakenteista käytettäväksi luokkakaaviota (Class Diagram), jonka avulla on helppo tutkia kuinka erilaiset rakenteet ovat sidoksissa toisiinsa (*kuva 4.9*). Luokkien suhteiden määrittelyn lisäksi voidaan tärkeimpien suhteiden välisistä toiminnoista muodostaa työkaaviot (Work Case View) (*kuva 4.11*), jotka sisältävät nimetyt työtehtävät ja niiden työnkulut.



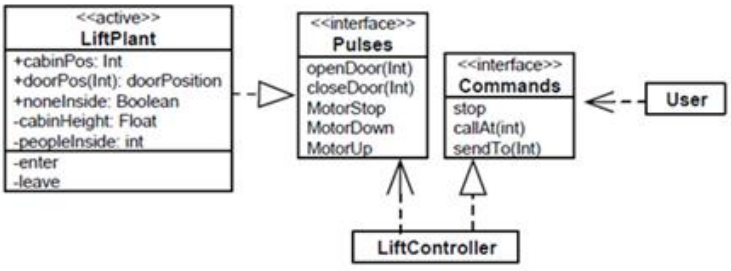
Kuva 4.11 *– Työkaavio*

Vaatimusten määrittelyssä tarkastellaan myös abstraktimpien asioiden liittymistä suunniteltavaan järjestelmään. Eli esimerkiksi kuinka muut ohjelmat tai järjestelmät liittyvät ohjelmaan ja millaisia erilaisia käyttäjäryhmiä ohjelmalla on. Näistä voidaan muodostaa käyttötapauskaavio (Use Case Diagram) (*kuva 4.12*), joka kertoo graafisesti millaisia käyttötapauksia erilaisilla käyttäjillä voi olla suunniteltavan ohjelman kanssa. Tarmon karkean tason käyttötapauskaavio on nähtävillä *kuvassa 2.7*.



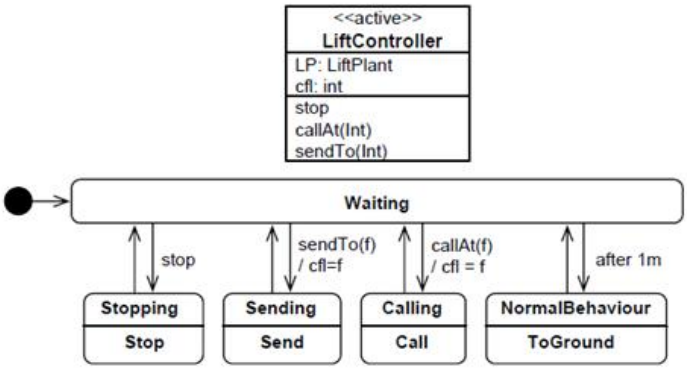
Kuva 4.12 *– Käyttötapauskaavio*

Tämän lisäksi vaatimusten määrittelyn vaiheessa voidaan tarkentaa luokkakaavioita (*kuva 4.13*), niin että ne sisältävät myös ohjelman sisäisen rakenteen elementtejä ja suhteita. Myös työkaavioita voidaan laajentaa ja lisätä niihin sisäisiä toimintoja selittäviä kuvaajia.



Kuva 4.13 *– Tarkennettu luokkakaavio*

Suunnittelun määrittelyssä suunnittelu viedään melko yksityiskohtaiselle tasolle. Tällöin otetaan huomioon muun muassa datan kuvaus (data view), joka kertoo mitkä ohjelman entiteetit käyttävät mitäkin datatyyppejä. Myös UML:n staattista kuvausta (aiemmin mainittu luokkakaavio) tarkennetaan ja siitä rakennetaan niin yksityiskohtainen, että se määrittelee jopa muotoja joilla dataa tulee tallentaa ja siirtää eri rajapintojen välillä. Choppy ja Reggio tuovat tässä vaiheessa esille myös käyttäytymiskuvauksen (behaviour view) (*kuva 4.14*), joka kuvaa kuinka staattisten rakenteiden luokat toimivat missäkin tilanteissa.



***Kuva 4.14** – Käyttäytymiskuvaus*

Choppyn ja Reggion artikkeli ei ota kovinkaan vahvasti kantaa itse kehitysprosessin kulkuun, vaan antaa enemmänkin pääpiirteiset suuntaviivat mitä pitkin voidaan kulkea, jos valitaan heidän esittelemänsä lähestymistapa. Sen sijaan he keskittyvät artikkelissaan kuvaamaan kuinka UML-pohjaisia työkaluja voidaan käyttää apuna eri vaiheissa kehitysprosessia. Artikkelin perusteella on selvää, että UML-työkaluja voidaan käyttää tehokkaasti avuksi, vaikka valittu kehitysprosessi ei noudattaisikaan artikkelin kolmivaiheista prosessia.

Choppyn ja Reggion artikkelissa esitetyn kolmiportaisen kehityksen kokonaisprosessin hataran käsittelyn takia Tarmon kehitysprosessia ei haluttu perustaa sen pohjalle. Sen sijaan Tarmon kehitysprosessin perustana käytetään myöhemmin tässä päätason kappaleessa esiteltävää Clean Wheel-prosessia. UML:n tuomia mahdollisuuksia tullaan kuitenkin käyttämään hyväksi osana valittua kehitysprosessia.

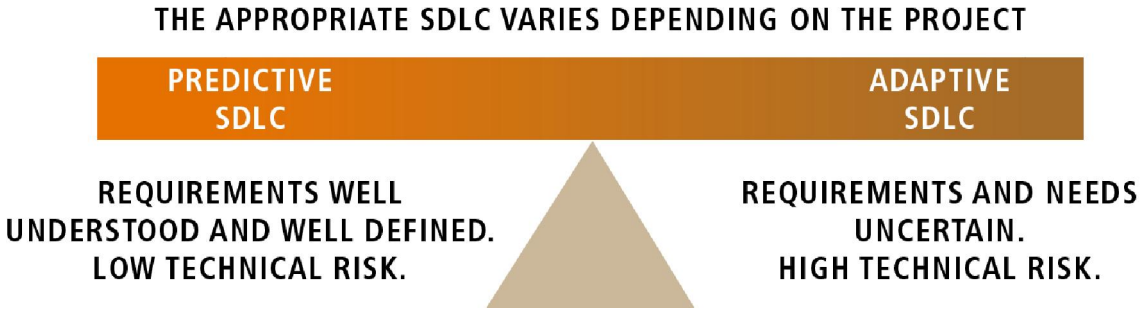
Osaksi Tarmon kehitystä ja myöhemmin esiteltävää Clean Wheel-prosessia ei nähty tarpeelliseksi käyttää kaikkia mahdollisia UML-kaavioita, vaan päädyttiin käyttämään viittä diagrammi-tyyppiä:

- *Luokkakaaviolla* tarkoitetaan kaaviotyyppiä, jolla kuvataan systeemin osia, sekä niiden staattisia relaatioita osana systeemiä. Tämän lisäksi luokkakaavio kertoo lohkon attribuuteista, toiminnallisuuksista ja niiden suhteesta alisysteemeihin.
- *Käyttötapauskaavio* näyttää systeemin aktorit, erilaiset käyttötapaukset ja toiminnot, joita aktorit voivat suorittaa. Tämän lisäksi kaavio näyttää aktorien väliset toiminnalliset suhteet ja linkitykset systeemin muihin toiminnallisuuksiin.

- *Tilakaavio* näyttää systeemin tilamuutokset, sekä tiettyjen tilanteiden saavuttamisen aiheuttamat tilamuutokset luokkatasolla.
- *Aktiviteettikaaviota* käytetään siihen, että voidaan analysoida tiettyjen valittujen käyttötapausten toimintaa ja nähdään kuinka eri osat systeemiä ovat sitoutuneet käyttötilanteeseen.
- *Sekvenssikaavion* tehtävänä on esittää väliaikaisten interaktioiden olemassaoloa aktoreiden ja objektien välillä, jotka tapahtuvat viestien kautta. Tämän avulla mallinnetaan systeemin käyttäytymistä karkealla tasolla, minkä avulla voidaan tarkastella kuinka yksittäiset käyttötapaukset ratkaistaan systeemitasolla.
  [Basile & Chicchio 2009]

4.3. System Development Life Cycle

Kun lähdetään miettimään millaista kehitysprosessia systeemin kehitykseen sovelletaan, kannattaa pohdinnan alle ottaa systeemin kehityksen elinkaari (System Development Life Cycle, SDLC) [Satzinger 2005]. *Kuvassa 4.15* on esitetty karkean tason muuttujat joiden perusteella SDLC-metodi voidaan valita systeemin kehitykselle sopivaksi. Mikäli kehitettävän systeemin vaatimukset ymmärretään hyvin ja ne on hyvin määritelty, sekä tekniset riskit ovat pieniä, voidaan valita ennalta tarkasti määritelty SDLC. Mikäli vaatimukset ja tarpeet ovat kuitenkin epävarmoja tai huonosti määriteltyjä ja tekniset riskit suuria, on SDLC-metodin oltava adaptiivinen, jotta se mukautuu systeemin elinkaaren aikana vastaan tuleviin muutoksiin.



Kuva 4.15 *– SDLC-metodin määritys*

Tarmon kehityksessä edellä esiteltyjen muuttujien tilanne on seuraava:

- *Vaatimukset* ovat osittain hyvin määritelty, mutta sisältävät myös huomattavia epävarmuuksia. Nykyisten osasysteemien osalta vaatimukset ovat melko selviä ja kohtalaisen hyvin määritelty. Toisaalta systeemin kehityksen vaatimuksena on myös se, että systeemin pitäisi elinkaarensa aikana kyetä sopeutumaan uusiin vaatimuksiin liiketoimintaympäristön muuttuessa. Tämä on seurausta siitä, että Metso kehittää jatkuvasti uusia konsepteja ja päivittää vanhoja, jolloin myös mitoitussysteemin on kyettävä muuttumaan dynaamisen liiketoimintaympäristön mukana. Ennalta tarkasti määritellyn SDLC:n staattisempi lähestymistapa söisi

systeemin elinkaaren liian lyhyeksi. Toisaalta puhtaasti adaptiivinen metodi tekisi systeemin kehityksestä liian orgaanista ja vaikeasti seurattavaa.

- *Tekninen riski* on kohtalaisen pieni. Systeemiä kehitetään Microsoft Excelin ja VBA:n pohjalle, koska ne ovat hyvin tuettuja ja aktiivisesti päivitettäviä järjestelmiä. Toisaalta kehitettävästä järjestelmästä on olemassa myös nykyinen ”hajautettu” toimintamalli, johon voidaan ongelmien kasautuessa tehdä ”roll-back”. Suurin riski teknisesti onkin niissä elementeissä, jotka eivät ole Metson käsissä. Esimerkiksi tilanteessa, jossa uusi mitoitussysteemi lyö itsensä läpi ja saavuttaa aseman jossa se on ainoa käytettävä järjestelmä käyttöjen mitoituksessa. Tällöin systeemin pohjana olevaan Exceliin tai VBA:han kohdistuva tietoturvapäivitys voi aiheuttaa ominaisuuksien toimimattomuutta. Tämä aiheuttaisi tilanteen, jossa mitoitusjärjestelmä muodostaisi mekaanisten käyttöjen tarjoustoiminnan pullonkaulan siihen asti kunnes Tarmo saadaan adaptoitua pohjajärjestelmän muutoksiin. Menneiden kokemusten perusteella tämäntyylisiä päivityksiä tulee, mutta ne ovat yleensä pieniä ja mitoitusjärjestelmän päivitykset saadaan tehtyä riittävän nopeasti, eikä niistä muodostu vakavaa riskiä.

Näiden perusteella valittiin kehitysprosessi, joka on ennalta määrätyn ja adaptiivisen SDLC:n välissä. Tarmon kehitysprosessiksi valittiin seuraavassa kappaleessa esiteltävä CleanWheel-prosessi.

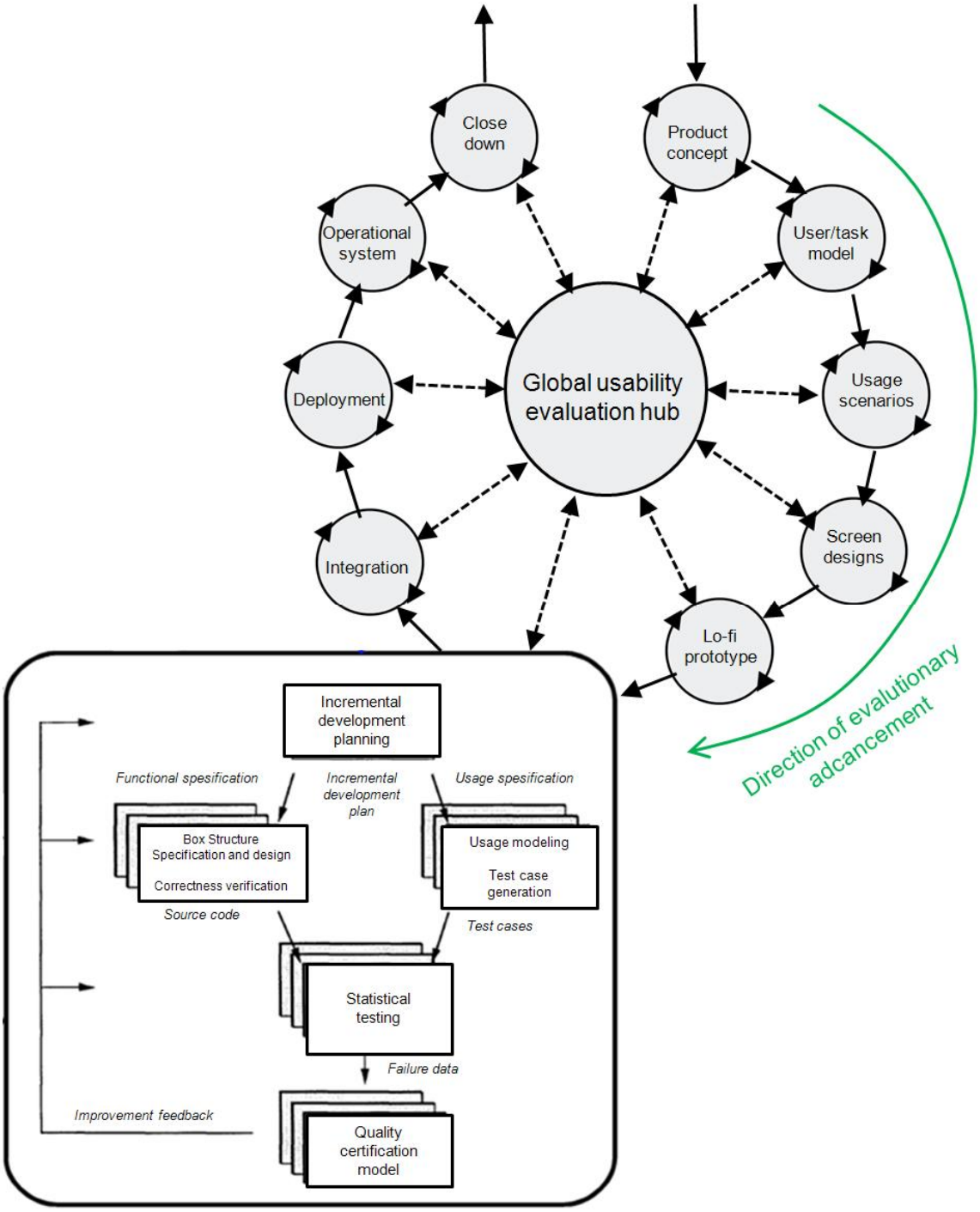
4.4. Clean Wheel - prosessi

Clean Wheel-prosessi on ”Suunnittelutyökalun käytettävyys ja kehitysprosessi”-kandidaatin työssä esitetty kahden kehitysprosessin fuusio [Immonen 2010]. Työn tavoitteena oli etsiä formaali prosessimalli erilaisten suunnittelutyökalujen kehittämistä varten. Työssä käsiteltiin suuri määrä erilaisia kehitysprosesseja ja niiden pohjalta muodostettiin Metson toimintaympäristöön optimoitu malli.

Kaikki työn teoriaosuudessa esitellyt kehitysprosessit ovat itsessään potentiaalisia vaihtoehtoja, mutta parhaan yhdistelmän tarjosivat ”The Wheel”-metodi ja ”Cleanroom software engineering”-prosessi. Ne ovat jo itsessään suunniteltu yhdistelemään suppeampien menetelmien parhaita puolia. Fuusio näistä kahdesta tarjosi parhaan kokonaisratkaisun, joka toimisi sekä vähän pienempien, että isompien työkalujen suunnitteluun, koska prosessi taipuu dynaamisesti erikokoisiin projekteihin, kun askeleita voidaan tarvittaessa pienempien työkalujen kehityksessä poistaa.

Kuvassa 4.16 on nähtävillä kaavio yhdistetystä Clean Wheel-prosessista. The Wheelistä tähän fuusioon periytyi erityisesti sen iteraatioaspektit, sekä käytettävyyden vahva läsnäolo. Cleanroom prosessista periytyi sen sijaan etenkin inkrementaalisen suunnittelun tuomat edut, sekä testaamisen tärkeyden korostaminen. Cleanroom osuus sijoittuu Clean Wheel-prosessissa prototyyppien (Lo-fi prototype, Hi-fi prototype) rinnalle/tilalle, koska Cleanroomin selkeä vahvuusalue on itse kehitysprosessissa, ei niin-

kään projektin valmistelevissa- ja jälkitoiminnoissa. Siksi nämä osuudet on jätetty pois Cleanroom-prosessista osana Clean Wheel – kokonaisprosessia.



***Kuva 4.16** – Clean Wheel-prosessikaavio*

Clean Wheel-prosessiin voidaan upottaa myös UML-suunnittelua, sekä moduuliarkkitehtuurin suunnitteluun tarkoitettuja prosesseja, jotka on esitetty aiemmin tässä kappaleessa. Myös kappaleessa ”5. Käytettävyys” käsiteltävät asiat ovat suuressa roolissa osana Clean Wheel-prosessia, koska käytettävyys on koko prosessin keskiössä.

Ennen Metsolla ei ole ollut yhtenäistä tapaa toteuttaa suunnittelutyökalujen suunnittelua, vaan prosessi on ollut tietyllä tapaa ”intuitiivinen”. *Kuvassa 4.17* eritellään kuinka Clean Wheel vastaa intuitiivisessa prosessissa tunnistettuihin ongelmiin.

Ongelma	**Clean Wheel - ratkaisu**
Prosessi ei tuottanut dokumentointia prosessista tai yksittäisistä päätöksistä.	Clean Wheelin yksittäisten pienten iteraatioiden osana tuotetaan dokumentaatiota tehdyistä ratkaisuista.
Prosessin vaiheita ei määritelty alussa lainkaan, joka aiheutti sen, että tekoaikataulua oli vaikea arvioida ohjelman tekemisen alkuvaiheissa (vasta projektin puolestavälistä eteenpäin pystyi karkeasti arvioimaan kokonaisaikataulua).	Clean Wheelissä on selkeät askeleet, jotka käydään läpi. Cleanroom osuuteen päästäessä voidaan tuleva aikataulu määritellä jo melko tarkasti, mutta jo ”Usage scenarios”-askeleessa voidaan arvioida projektin kokonaiskestoa.
Projektin alussa ohjelman tulevilla käyttäjillä ei ollut selkeää näkemystä millaisen ohjelman he halusivat ja näin ollen tavoitteet olivat hieman epäselvät. Tämä aiheutti ylimääräistä iterointia niin käyttöliittymän kuin ohjelman ytimen toiminnan suhteen.	Clean Wheel määrittelee projektin alkuun selkeitä askeleita, joilla pyritään selvittämään mitä asiakkaat/käyttäjät oikeasti haluavat. Näin prosessi pakottaa käyttäjät miettimään mitä työkalulta halutaan ja turhalta iteroinnilta vältytään.
Käyttöliittymästä olisi voinut tulla parempi.	Clean Wheel ottaa aktiivisesti kantaa käytettävyyteen koko prosessin ajan, jolloin käytettävyyden huomiointi on avainasemassa.

***Kuva 4.17** – Clean Wheel vs. intuitiivisen prosessin ongelmat [Immonen 2010]*

Luottamusta Clean Wheelin-prosessin käyttöön osana Tarmon kehitystä antaa se, että Clean Wheel-prosessia on jo onnistuneesti käytetty osana erästä toista suunnittelutyökalun kehitysprosessia Metson sisällä. Tuo projekti tuotti toimivan ja erikseen käyttäjiltä kiitosta keränneen suunnitteluohjelman, joka on tälläkin hetkellä päivittäisessä käytössä.

4.5. Konfigurointi

Konfigurointi on Tarmon kannalta tärkeä käsiteltävä asia johtuen Metson projektiluonteisesta toiminnasta. Konfiguroinnin tunnusmerkkejä ovat muun muassa seuraavat [Lehtonen 2011]:

- Jokainen toimitus on yksilöllinen ja tehty asiakastilauksena.
- Toimitusprosessiin sisältyy ainoastaan rutiininomaista systemaattista muunnelmasuunnittelua.

- Tuoteyksilöt koostuvat ennalta suunnitelluista osista ja osien muodostamista kokonaisuuksista.
- Toimitettavia tuoteyksilöitä vastaa ennalta suunniteltu yleinen tuoterakenne.

Näistä neljästä tunnuspiirteestä täyttyy Metson tapauksessa täydellisesti kolme ja neljäskin osittain. ”Toimitusprosessiin sisältyy ainoastaan rutiininomaista systemaattista muunnelmasuunnittelua”-kohta ei täyty kokonaan, koska useissa projekteissa on mukana myös pieniä määriä epärutiininomaista suunnittelua. Tämä johtuu siitä, että asiakkaiden toiveet ovat hyvin erilaisia ja paperikoneen sijoituspaikat saattavat asettaa omanlaisiaan vaatimuksia niin koneen prosessille, turvallisuusasioille kuin rakenteelle yleensäkin. Suurin osa näistä projektikohtaisista muunnelmasuunnittelusta on kuitenkin pyritty systematisoimaan toiminnan tehostamiseksi.

Kun konfiguroinnilla pyritään asiakasvarioituviin tuotteisiin, voidaan perustellusti kysyä, että puhutaanko samasta asiasta kuin moduloinnissa? Tietyllä tapaa kyllä, mutta konfigurointi tulee mieluummin ajatella moduloinnin yläkäsitteeksi. Konfigurointi on liiketoimintamalli, jonka toteuttamisessa käytetään hyväksi modulaarisuutta. Konfiguroinnin käyttö onkin tietyllä tapaa strateginen päätös, jonka tekeminen vaatii muutamia asioita, jotka on esitetty *kuvassa 4.18*.

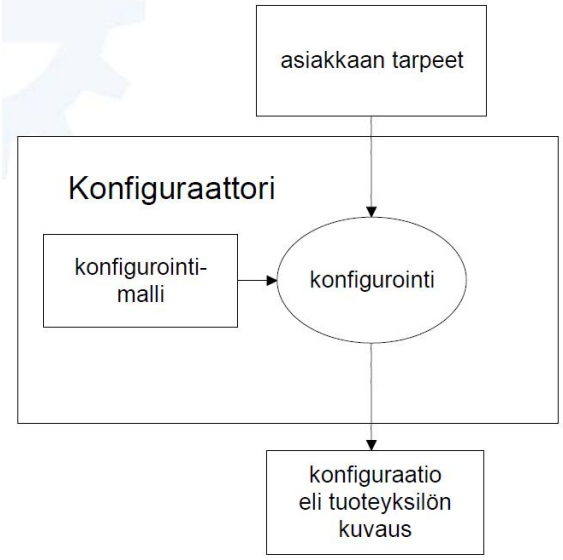
Vaatimus	**Metson tapauksessa**
Toistoja on riittävästi	Isoja projekteja on kymmeniä vuodessa ja pienempiä projekteja tätäkin enemmän.
Tarve laajaan valikoimaan	Rakenneryhmiä on seitsemän ja jokaisessa rakenneryhmässä rakenneryhmäkohtaisia konsepteja on lukuisia. Konseptien sisällä on myös laajat määrät asiakasvarioituvia komponentteja. Kokonaisuuden variaatiovalikoima on suuri.
Asiakastarpeet riittävän samanlaisia, jotta modulaarinen yhdistely on mahdollista.	Paperin tekeminen on jo vanha prosessi, joten peruskonseptit ovat hyvin vakioituneita ja näin ollen myös asiakastarpeet riittävissä määrin samanlaisia.

***Kuva 4.18** – Konfigurointipäätöksen tukiehdot*

Konfiguroitavat tuoteperheet sopivat yrityksille joissa on systemaattinen toimintakulttuuri ja organisaatiomalli, joka tukee konfigurointia. Konfiguroinnin valinta osaksi tehokasta strategiaa vaatii hyvän ymmärryksen yrityksen markkinoista ja tuotteista. Tällöin yrityksen pitää tietää asiakkaiden vaatimukset ja toiveet, sekä sen tulee kyetä tuottamaan markkinoiden vaatimuksia vastaavia tuotekonsepteja. Mikäli tuntemusta ei ole riittävästi, joudutaan jatkuvasti räätälöimään asiakastarpeiden mukaisesti, joka ei ole konfiguroinnin tarkoitus. [Lehtonen 2011] Metsolla on ”Paperit”-liiketoimintalinjan

puolella vielä työtä tehtävänä, jotta konelinjat saataisiin aidosti konfiguroiduiksi. Tällä hetkellä asiakasvariointia ja räätälöintiä joudutaan vielä tekemään ”turhan” paljon.

Konfigurointi menetelmänä vaatii komponenttien valintaa olemassa olevasta komponenttipoolista. Komponenttien ei kuitenkaan tarvitse olla täysin staattisia, vaan ne voivat olla vaihtoehtoisia, lisävarusteita tai parametrisia. Jotta tämä vaatimus voidaan toteuttaa, täytyy komponenttien välillä olla määriteltynä relaatiot (vrt. modulaarisuuden rajapinnat). Relaatioita voi olla erityyppisiä: abstrakteja, loogisia, topologisia tai komponentti-kustomoituja relaatioita. [Lehtonen 2011]



***Kuva 4.19** – Konfiguroinnin toimijat [Lehtonen 2011]*

Konfiguroinnin käyttö osana yrityksen toimintaa edellyttää konfigurointimallin olemassaoloa. Tällä mallilla kuvataan tuoteperheen relaatiot: mikä komponentti/moduuli sitoutuu mihinkin, mitkä ovat keskenään vaihtokelpoisia, mitkä ovat kokonaan optionaalisia, mitä niistä voi yhdistellä, millä perusteilla valinnat tehdään, millaisia parametreja komponenteilla on, mikä niiden toiminnallisuus on jne. *Kuvassa 4.19* nähdään kuinka konfigurointimalli sijoittuu konfigurointiin kokonaisuutena. Koko konfigurointiprosessi lähtee liikkelle asiakastarpeista. Asiakastarpeet tuodaan konfiguraattoriin, joka sisältää sekä konfigurointimallin, että konfiguroinnin itsensä. Näiden tietovirtojen perusteella muodostuu lopputuloksena tuoteyksilön konfiguraatio.

Konfiguraatiomallin rakenne voidaan rakentaa kahdella päätavalla: joko konfiguraattorin omaan sisäiseen malliin tai vaihtoehtoisesti esimerkiksi osaksi PDM-järjestelmää. Rakennemallista tulee kuitenkin tulla esille järjestelmän osaset, attribuutit, sekä relaatiot. Sääntöjen rakentaminen vaatii yleensä ehtolauseita ja syy-seurauskaavioita. Mallia rakentaessa tulisi miettiä myös miten esimerkiksi kustannusmallit liittyvät konfiguraattoriin ja millaisen rajapinnan kautta. Konfiguraattorin rakentamisessa tulee huomioida myös erilaiset käyttäjävaatimukset: mikäli on tiedossa että konfiguraattoria käyttävät vain pitkän linjan asiantuntijat, voidaan konfiguraattorista rakentaa monimutkaisempi, joka jättää enemmän käyttäjänsä vastuulle. Mikäli käyttäjä-

kunnassa on myös ihmisiä, joita ei voi pitää asiantuntijoina, tulee myös konfiguraattori rakentaa siten, että se on riittävän yksinkertainen ja ohjaa käyttäjää mahdollisimman hyvin. Tämä asettaa suurempia vaatimuksia rakenteelle ja käyttöliittymälle – yleisesti koko toteutusprosessi on tällöin raskaampi.

Konfiguraattorien rakentamisessa on kuitenkin ongelmia, jotka on syytä pitää mielessä, jotta niihin ei kompastuttaisi. Konfiguraattorin vaatiman konfigurointimallin sisältämät relaatiot ja säännöt muodostuvat helposti huomattavan monimutkaisiksi, joka tekee mallin ylläpidosta haastavaa. Tämä johtaa toiseen tyypilliseen ongelmaan: lyhyeen elinikään. Koska ylläpito on haastavaa, jää konfiguraattori helposti yrityksen kehityksen tahdista ja vanhentuneena se ei enää vastaa niihin tarpeisiin, joihin se alun perin rakennettiin. Näihin ongelmiin voi tehokkaimmin vastata sillä, että pyritään tekemään mallista selkeä ja riittävän yksinkertainen, jaetaan ylläpitovastuut pienempiin osiin ja nimetään vastuuhenkilöt/-osastot selvästi. Tämän lisäksi tulee pitää huolta, että ensin katsotaan yrityksen tuoteperhe kuntoon, jotta tiedetään mihin tarpeeseen konfiguraattorin pitää vastata. Mikäli tuoteperheiden määrittelyt ja keskinäiset suhteet eivät ole alun perin kunnossa tulee toteutusvaiheesta hidas ja helposti käy niin, että konfiguraattorin tekijät joutuvat linjaamaan relaatioita osana tekemistä, jolloin väärinkäsitykset ja virheet ovat hyvin todennäköisiä.

Yhteenvetona konfiguroinnista on syytä todeta, että mikäli konfigurointi valitaan yrityksen strategiaksi ja rakennetaan konfiguraattori, niin järjestelmän rakentamiseen ja ylläpitoon on uskallettava investoida riittävästi ja pitkällä aikajänteellä. Pienellä investoinnilla ja lyhyellä aikajänteellä toteutettaessa konfigurointi teettää vain ongelmia.

4.5.1. Konfigurointi ja Tarmo

Konfiguroinnin käsittely Tarmon kaltaisen systeemin ohessa on perusteltua, koska Tarmon käyttöliittymä itsessään sisältää tietynlaisen konfiguraattorin. Sillä määritellään millainen paperikoneen rakenne on, jonka perusteella voidaan määritellä komponentit ja näiden avulla selvittää tehonkulutukset ja jakaa ne käyttöryhmän moottoreille. Toinen tärkeä syy käsitellä konfiguraattoria osana järjestelmän kehitystä on kustannusmallien integrointi osaksi konfiguraattoria (Tarmoa). Tarmoon integroituu mekaanisen välitysketjun kustannusmalli, jolla hinnoitellaan vaihteet, kytkimet ja perustustarvikkeet. Rajapintana tähän laskelmaan toimii Tarmon tuottama teholista.

Metsossa on pyöritelty ideatasolla myös laajempaa konfiguraattoria, joka sisältäisi ”koko” paperikoneen ja tuottaisi tuloksena ”äiti-dokumentin”, jota voitaisiin hyödyntää lukuisissa muissa mitoitus- ja kustannusmalleissa – myös Tarmo voisi hyödyntää ”äiti-dokumenttia”. Näin laajan konfiguraattorin tekemiseen ei kuitenkaan ole vielä löytynyt riittävästi tahtoa, eikä investointihalukkuutta. Ymmärrettävästi mallin ylläpitotyö olisi haastavaa paperikoneen suuren nimikemäärän takia (tuhansittain nimikkeitä). Toistaiseksi projektin riskit nähdäänkin suuremmiksi kuin hyödyt, joten mahdollinen ”äiti”-konfiguraattori odottaa toistaiseksi tulemistaan.

4.6. Systeemien käytettävyys

Käytettävyys on saanut ohjelmistokehityksessä viime vuosina yhä enemmän huomiota, kun on huomattu, että se on tärkeämpi arvoa tuottava ominaisuus asiakkaalle kuin on aiemmin osattu ajatella. Kärjistetysti voidaankin sanoa, että järjestelmän hyvyys laajemmin ajateltuna heijastuu käyttäjälle käyttöliittymän ja käytettävyyden läpi. Heikosti käytettävä, mutta taustoiltaan hyvä järjestelmä voi heijastua käyttäjälle heikompana kokonaisuutena kuin hyvin käytettävä, mutta taustoiltaan heikko järjestelmä.

Artikkelissa ”A methodology and tools for applying contex-specific usability guidelines to interface design” [Henninger, 2000] esitellään asioita, joita käytettävyyden suunnittelussa kannattaa huomioida. Seuraavassa on listattuna tiivistetysti tärkeimpiä artikkelissa esiteltyjä asiakokonaisuuksia [Immonen 2010]:

- Muodostetaan sääntöjä/ohjeita designin luomiseksi ja näin voidaan tulevissa projekteissa käyttää hyväksi aiempaa kokemusta.
- Ihmisen ominaisuuksia ymmärtävän spesialistin varhainen mukanaolo suunnitteluprosessissa voisi korjata huonot suunnitelmat ennen kuin ne ehtivät muodostumaan. Tämän huomioinnissa suurin ongelma on jatkuvasti lyhenevät kehitysajat.
- Suunnittelusäännöt eivät automaattisesti vähennä huonoja sovelluksia suunnittelusta, koska ihmiset tulkitsevat näitä sääntöjä omalla tavallaan.
- Suunnittelusääntöjen on syytä olla konkreettisia, eikä abstrakteja.
- Väestön ja maiden erojen vuoksi on mahdotonta luoda standardi-ohjetta, jota voitaisiin hyödyntää jokaiseen palveluun/systeemiin.
- Sääntöjen tulee vastata organisaation omiin tarpeisiin
- Suunnittelijat etsivät konkreettisia malleja käyttöliittymillä, jotka ovat todistetusti toimivia ja omaksuvat niistä toimintamalleja omiin suunnitelmiinsa (toimii hyvin, jos käyttäjillä on samanlaiset taustat ja tarpeet).
- Yritysten kannattaa muodostaa omat säännöt, joita sekoitetaan yleisiin sääntöihin. Tällöin voidaan hyödyntää yrityksen hiljaista sisäistä tietoa.
- Riskinä on, että säännöt laajenevat/eroavat niin paljon, ettei löydy yhteistä linjaa suunnitelman laatimiselle, jotta käytännöllinen, oikeasti tehokas sääntöpaketti saataisiin tehtyä.

Gulliksen, Boivie ja Göransson käsittelevät artikkelissaan "Usability professionals—current practices and future development" käytettävyyden suunnittelun nykytilaa ja visioivat mahdollisia tulevia kehityssuuntia. Artikkelissa listataan asioita joita käytettävyyden suunnittelijoiden (usability designer, UD) tulisi huomioida.

Kirjoittajien mielestä huono käyttöliittymän (Human Computer Interface, HCI) suunnittelu voi edistää työkyvyttömyyttä ja tehottomuutta. Tällaiseen ilmiöön syitä voi olla lukuisia, mutta artikkeli korostaa tärkeänä syynä työmuistin turhaa kuormitusta. Tämä voi aiheuttaa sen, että käyttäjän keskittyminen herpaantuu lukuisista pienistä,

”turhanpäiväisistä” tehtävistä ja aiheuttaa näin virheitä, sekä stressaantumista. Tähän ratkaisuksi ehdotetaan automatisoinnin viemistä niin pitkälle kuin mahdollista niin yksittäisten tehtävien kuin isompienkin kokonaisuuksien osalta. Käyttäjän tehtäväksi pitäisi jättää ainoastaan sellaiset asiat, joita ei automatisoinnilla voida suorittaa. Näin tekijä ei koe tekevänsä turhaa työtä, vaan kokee oikeasti voivansa vaikuttaa isompiin kokonaisuuksiin. Tämä toimii hyvänä motivaattorina ja näin ollen myös tehokkuuden, sekä työn mielekkyyden lisääjänä.

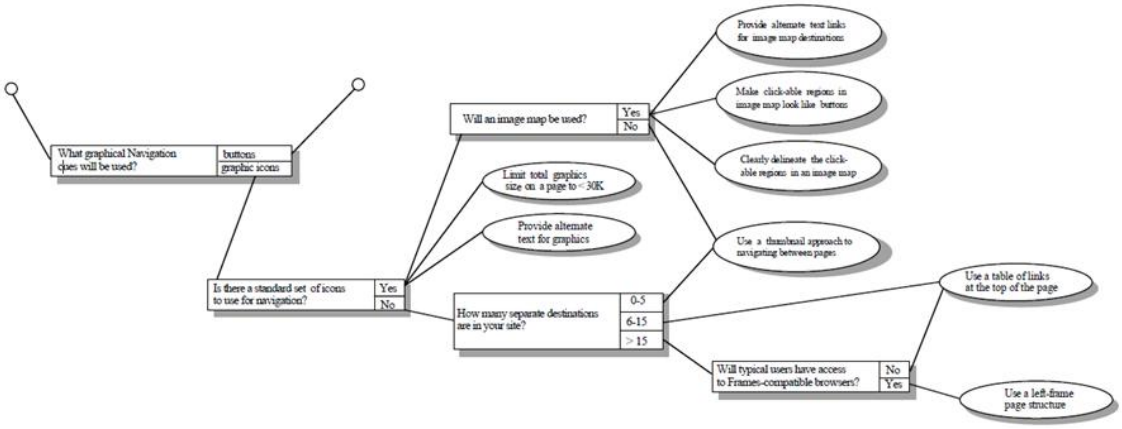
Artikkelin [Gulliksen, Boivie, Göransson, 2006] tutkimuksen mukaan käytettävyyden ammattilaisia ei ole tarpeeksi kehitysprojektien avainasemissa ja tämän takia käytettävyyden asiat, sekä user-centred system design (UCSD) eivät saa tarvitsemaansa painoarvoa. Käytettävyyden ammattilaiset ovat usein alistettuina työryhmiin eikä tiimien vetäjillä ole tarpeeksi usein tarpeeksi näkemystä käytettävyyden asioiden osalta, jotta osaisivat arvostaa UD-ihmisten panosta. Siksi UCSD:tä toteutetaan liian usein vasta muiden asioiden jälkeen, ikään kuin päälle liimattavana asiana, vaikka sen tulisi olla mukana alusta asti kiinteänä osana kehitysprosessia. Tämä arkkitehtuuriin keskittyvä lähestymistapa on kuitenkin onneksi vähitellen muuttumassa ja prosessiin sidotaan käytettävyysasiat yhä aikaisemmin. Toisaalta myös käyttäjät pitää tuoda osaksi kehitysprosessia, koska juuri heillä on avaintietämys mitä tuotteelta tai ohjelmalta vaaditaan, että se on helppo ja tehokas käyttää. Tämän lisäksi kaikki käyttäjätasot tulisi huomioida kun suunnitellaan käytettävyyttä.

Kun tuotekehitysprosessia halutaan viedä UCSD:n suuntaan ja ymmärrystä käytettävyydestä lähdetään rakentamaan, täytyy välttää sitä, että asia esitellään vain pintapuolisesti. Tällöin käytettävyydestä jää helposti kuva vain koristeellisena asiana ja sen syvällisempi ymmärrys ja tärkeys jäävät tekijöiltä ymmärtämättä.

Kirjoittajien mielestä suurimpia hidasteita käytettävyyden läpilyöntiin kiinteäksi osaksi tuotekehitysprosessia on muutosvastarinta, tietämättömyys, aika, resurssit ja osaajien puute. Muutosvastarinta ja tietämättömyys liittyvät kiinteästi toisiinsa, joten käytettävyyden ymmärtämisen lisääminen vähentää myös vastustusta. Ajan puute periytyy puolestaan jatkuvasti tiukentuvista kehitysaikatauluista, jolloin toissijaisena pidettävät käytettävyyden asiat helposti ovat ensimmäisinä karsittavien tai ainakin vähemmän huomiota saavien asioiden listalla. Resurssit ja osaajien puute johtuu puolestaan siitä, että tällä hetkellä käytettävyyden asioita ei juuri voi opiskella ja näin ollen käytettävyyden asiantuntijat ovat vielä tänäkin päivänä yleensä enemmän tai vähemmän itse oppineita.

Artikkeli listaa myös ominaisuuksia mitä käytettävyyssuunnittelijan (UD) tulisi hallita, jotta tämä voisi hoitaa työnsä menestyksekkäästi. Tämä listaus antaa kuvaa siitä kuinka laajaan kenttään käytettävyys itse asiassa pureutuu. Listauksen mukaan UD:n tulisi omata hyvä perustietämys HCI:stä, sekä ihmisen fysiikasta, psyykestä ja sosiaalisuudesta. Tämän lisäksi UD:llä tulisi olla tarpeeksi ohjelmistosuunnittelun tietämystä, että ymmärtää kuinka UCSD asiat voidaan huomioida itse ohjelman teossa ja näin auttaa muita tekijöitä konkreettisilla neuvoilla. UD:n on omattava kehitystyökalujen tuntemusta, jotta hän voi ymmärtää mitä rajoitteita ne asettavat käytettävyyden huomioonot-

tamiselle ja toisaalta, mitä ne mahdollistavat. Myös tiedon hankinnan ja analysoinnin tuntemus on tärkeää, kuten esimerkiksi haastattelutekniikat, sisältöanalyysit, havaintotutkimukset yms. Nämä auttavat kokonaisuuden, sekä yksittäisten asioiden ymmärtämisessä. Poikkitieteellinen tuntemus eri aloista auttaa UD:ta myös selviämään tehtävästään, koska sen ansiosta hän voi ymmärtää eri alojen välisiä suhteita. Esimerkiksi talous, antropologia, tietotekniikka ja psykologia ovat helposti UD:n työssä hyödynnettäviä tieteenaloja. Näiden kaikkien lisäksi UD:n tulisi osata myös suunnitella, jotta voi samaistua muiden suunnittelijoiden työhön ja tarvittaessa itse suunnitella ja toteuttaa ideoita.



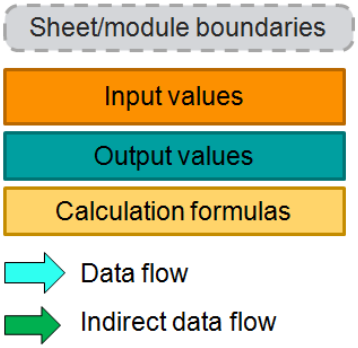
Kuva 4.20 *– Esimerkki yksittäisestä käytettävyyden asian päätösprosessista [Henninger 2000]*

Suunnittelun kannalta käyttävyysasioiden pohtimisen ei tarvitse olla intuitiivista ja epäformaalista. Henningerin artikkelissa [Henninger, 2000] esitetään yksi esimerkki yksittäisen käytettävyyden ongelmanratkaisuprosessista (*kuva 4.20*). Tämän tyyppisellä formuloidulla ja konkreettisella menetelmällä saadaan tehtyä päätöksiä siten, että jälkikäteen voidaan tarvittaessa tarkastella ratkaisuun johtaneet askeleet. Näitä yksittäisiä ratkaisuja voidaan myös uusiokäyttää, joka helpottaa suunnitteluprosessin pitkäjänteistä kehitystä ja tehostamista.

Tarmon kehityksessä tässä kappaleessa esitettyihin asioihin on pyritty kiinnittämään huomiota Clean Wheel-kehitysprosessin kautta. Clean Wheel-prosessi mahdollistaa käytettävyyden ja käyttäjäkeskeisyyden korostamisen. Käytettävyyttä sivutaan myöhemmin myös kappaleessa ”6.2 Ajatuksia käyttöliittymästä”.

5. RAKENNERYHMÄKOHTAISET LASKENTA-POHJAT

Olemassa olevien laskentapohjien analysointi pitää suorittaa, ennen kuin voidaan suunnitella laskentamoduulien rajapintoja ja integraatiota Tarmoon. Tässä kappaleessa käydään karkealla tasolla läpi olemassa olleiden laskentapohjien ominaisuuksia ja etsitään niistä parhaat ominaisuudet, joita voidaan jatkossa implementoida kaikille moduuleille osana laskentapohjien normaalia päivitystyötä.



***Kuva 5.1** – Selitteet moduulien periaatekaavioille*

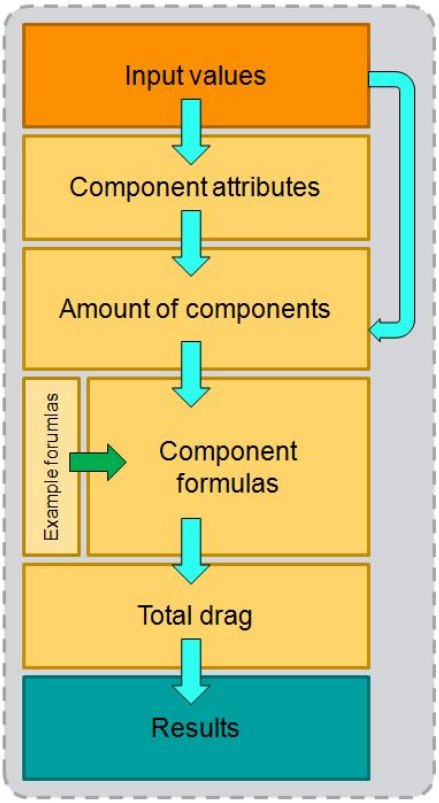
Kuvassa 5.1 on nähtävissä selitteet elementeistä, joita käytetään kunkin rakenneryhmän toimintaa kuvaavissa periaatekaavioissa. Erilaisten lähestymistapojen vertailun helpottamiseksi kaikissa kaavioissa käytetään vastaavia merkintöjä. Sivujen rajat, sisääntuloarvot, ulostulosarvot ja laskentakaavat ovat itseään selittäviä. Datavirralla (data flow) tarkoitetaan välitulosten, laskenta-arvojen, vakioiden tai muun informaation virtaa eri osioiden välillä. Epäsuora datavirta (indirect data flow) tarkoittaa puolestaan datavirtaa, jota ei käytetä laskennassa suoranaisesti, mutta epäsuorasti sen kautta siirtyvää tietoa käytetään hyväksi laskennassa tai ylläpidollisissa asioissa.

5.1. Viira-, puristin- ja kuivatusosa

Tarmon nykyiset laskentapohjat seuraavat samanlaista periaatemallia (*kuva 5.2*). Siinä laskenta jakautuu selkeästi erillisiin kokonaisuuksiin:

- Syöttöarvoihin
- Komponenttien ominaisuuksien määrittelyyn
- Komponenttien määrän analysointiin
- Komponenttikohtaiseen teholaskentaan

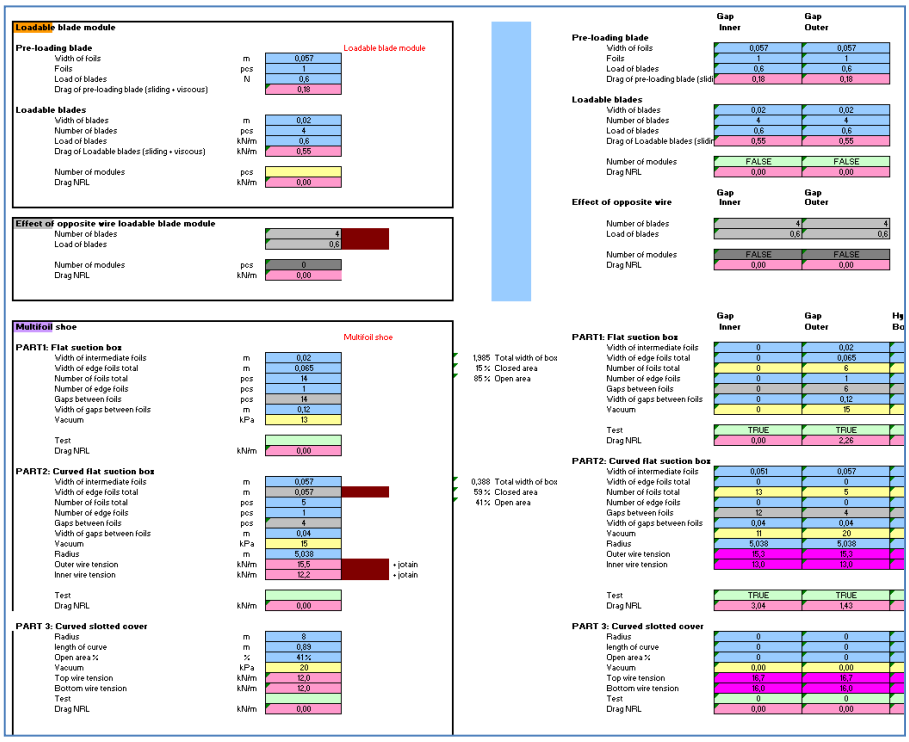
- Kokonaistehon määrittelyyn
- Lopputuloksiin



Kuva 5.2 *– Tarmon laskentapohjan toiminnan periaatekaavio*

Viiran, puristimen ja kuivaimen laskennan suurin ero muihin laskentapohjiin on se, että kokonaisuus on haluttu pitää selkeästi yhdessä paketissa yhden sivun sisällä. Tähän osaltaan vaikuttaa vanhan Tarmon toteutustapa, jossa käyttöliittymäsivut olivat kiinteä osa laskentapohjaa. Tällöin oli luontevaa, että laskenta tapahtuu yhdellä laskentasivulla, kun laskennan rajapinta on jaettu muille sivuille. Vaikka laskenta tapahtuukin yhdellä sivulla, on se silti jaettu useampaan toisistaan selvästi eroavaan palaseen. Kuten kuvaajasta voidaan nähdä, on laskenta virtaviivaisesti etenevä. Jokainen välivaihe käyttää hyväkseen edellisestä vaiheesta periytyviä tietoja. Laskennassa on viisi vaihetta, joiden perusteella päädytään lopullisiin käyttöpistekohtaisiin tehotietoihin.

Toinen erikoisuus muihin pohjiin verrattuna yllä esitetyssä periaatekaaviossa on komponenttikohtaisten kaavojen käsittely. Nämä kaavat on rakennettu siten, että jos kaavoja on tarve päivittää, se onnistuu helposti siten, että käyttäjä muuttaa ja testaa uudet kaavat esimerkkikaavojen kohdassa (example formulas). Kun kaavat on saatu toimimaan halutulla tavalla, voi esimerkkikaavan kopioida helposti koskemaan kaikkia komponenttikohtaisia valintoja ja laskukaavoja. Tämä tekee ylläpidollisista laskentapäivityksistä helppoja implementoida.



Kuva 5.3 *– Tarmon laskentapohjat*

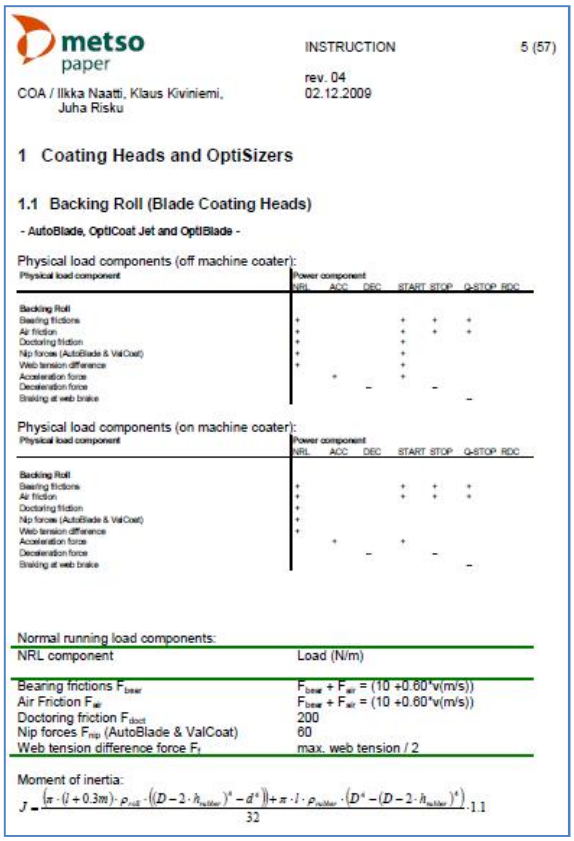
Kuvassa 5.3 on kuvankaappaus viiran, puristimen ja kuivaimen laskennasta. Tässä kuvassa on nähtävillä vasemmalla laatikoiden sisällä edellä kuvattu esimerkkilaskenta. Oikealla nähdään taas efektiivinen laskenta, johon päivitetyt esimerkkikaavat voidaan kopioida. Laskenta on laajalti värikoodattu, jotta erilaiset muuttujat ja laskennan osa-alueet on helpompi visuaalisesti tunnistaa.

Tarmon laskentapohjien avainomaisuuksia (vanhassa kontekstissaan):

- Käyttöliittymä on eriytetty rakenneryhmittäin: rakenneryhmäkohtaisessa laskentapohjassa on sekä laskenta, sekä kyseisen rakenneryhmän käyttöliittymä.
- Pääohjelma hoitaa tallennukset, avaamiset ja printit
- Ainoastaan printti siirtyy rakenneryhmän laskentapohjasta pääohjelmaan
- Pääohjelmasta rakenneryhmiin siirtyy ainoastaan kaikille rakenneryhmille yhteisiä päätietoja (nopeus, leveys yms)
- Laskennassa ei ole käytetty laskenta-ydin-ajatusta, jossa kaikki komponentit kierrätettäisiin yksittäisten laskentaytimien läpi, vaan kaikki laskenta tapahtuu samaan aikaan rinnakkain omien laskentojensa läpi.
- Laskentojen rakentamisessa on pyritty hyödyntämään ”palikoita”, joita monistetaan sitten matriisiksi, jotta kaikki komponentit saadaan käytyä läpi

5.2. Päällystysasema

Kuvassa 5.4 on kuvankaappaus ns. mitoitusohjeesta päällystysasemien osalta. Kaikilta rakenneryhmiltä löytyy vastaavat ohjeistukset, mutta ne eivät tyypillisesti ole aivan samanlaisessa formaatissa. Laskentapohjat sisältävät kuitenkin myös sellaista laskentatietoa ja -periaatteita, joita ei löydy mistään ohjeistuksesta yksiselitteisesti.



metso paper

INSTRUCTION 5 (57)

rev. 04
02.12.2009

COA / Ilkka Naatti, Klaus Kiviniemi, Juha Risku

1 Coating Heads and OptiSizers

1.1 Backing Roll (Blade Coating Heads)

- AutoBlade, OptiCoat Jet and OptiBlade -

Physical load components (off machine coater):

Physical load component	Power component NRL	ACC	DEC	START	STOP	Q-STOP	RDC
Backing Roll							
Bearing frictions	+			+	+	+	
Air friction	+			+	+	+	
Doctoring friction	+			+			
Nip forces (AutoBlade & ValCoat)	+			+			
Web tension difference	+			+			
Acceleration force		+		+			
Deceleration force			–		–		
Braking at web brake						–	

Physical load components (on machine coater):

Physical load component	Power component NRL	ACC	DEC	START	STOP	Q-STOP	RDC
Backing Roll							
Bearing frictions	+			+	+	+	
Air friction	+			+	+	+	
Doctoring friction	+						
Nip forces (AutoBlade & ValCoat)	+						
Web tension difference	+						
Acceleration force		+		+			
Deceleration force			–		–		
Braking at web brake						–	

Normal running load components:

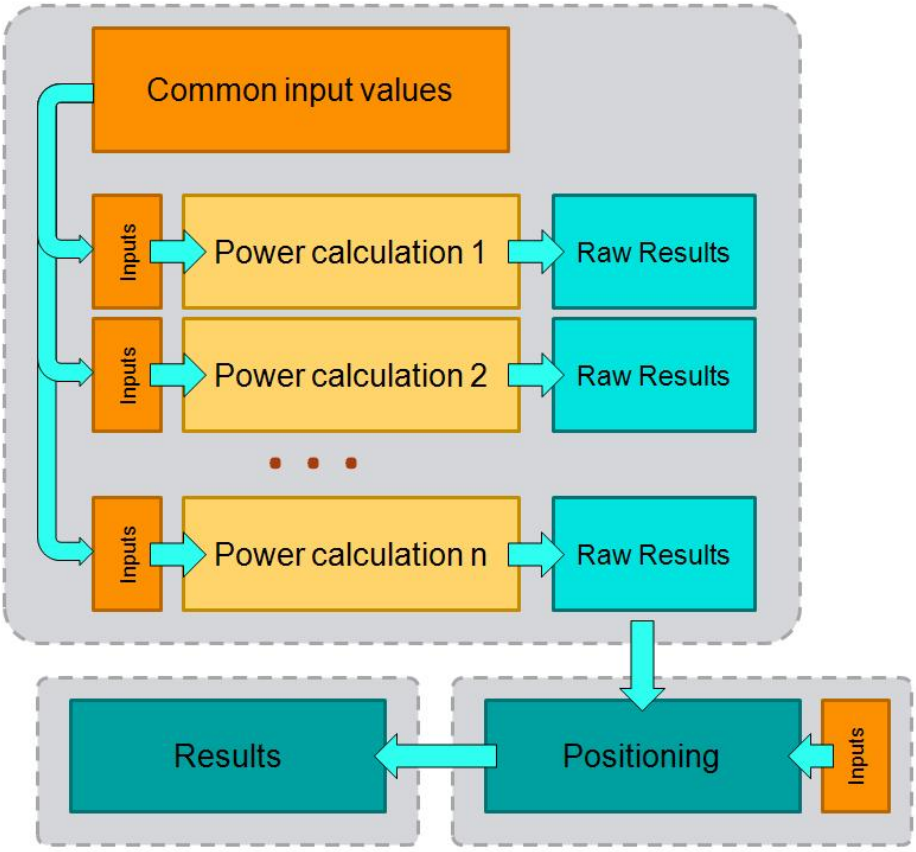
NRL component	Load (N/m)
Bearing frictions F_{bear}	$F_{bear} + F_{air} = (10 + 0.60 * v(m/s))$
Air Friction F_{air}	$F_{bear} + F_{air} = (10 + 0.60 * v(m/s))$
Doctoring friction F_{doct}	200
Nip forces F_{nip} (AutoBlade & ValCoat)	60
Web tension difference force F_t	max. web tension / 2

Moment of inertia:

$$J = \frac{\left(\pi \cdot (l + 0.3m) \cdot \rho_{roll} \cdot \left((D - 2 \cdot h_{coMer})^4 - d^4\right)\right) + \pi \cdot l \cdot \rho_{coMer} \cdot \left(D^4 - (D - 2 \cdot h_{coMer})^4\right)}{32} \cdot 1.1$$

***Kuva 5.4** – Päällystysaseman tehomitoitusohjeet (application instructions)*

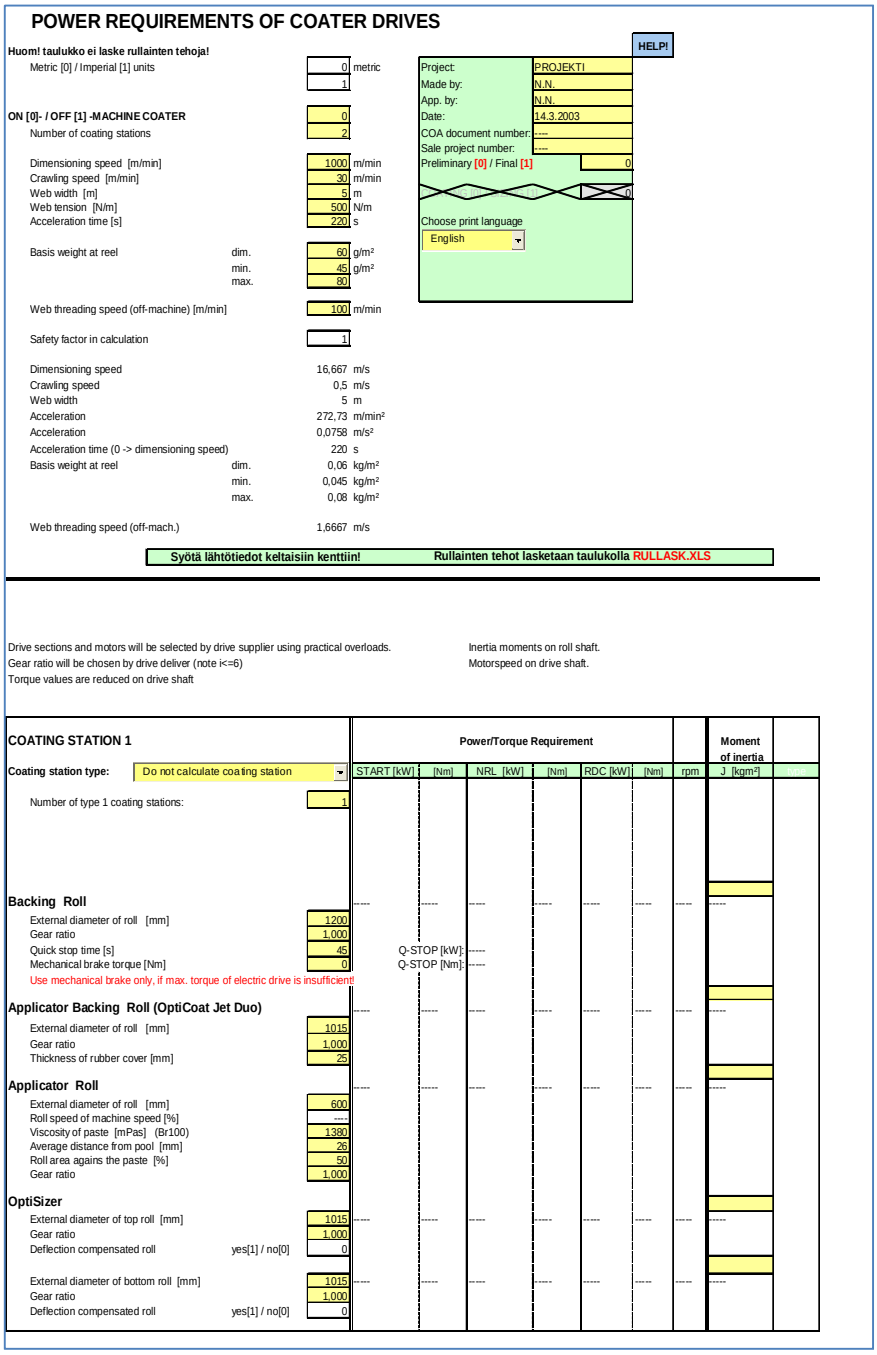
Jos applikointiohjetta päivitetään, niin sama päivitys tulee tehdä myös laskentapohjaan. Tällä hetkellä kun laskentapohjien ylläpitovastuut ovat osittain häilyviä, on ohjeiden ja laskentapohjien päivitysten välissä joskus pitkäkin viive. Tämä ongelma pyritään ratkaisemaan linjalaajuisella laskentaintegraatiolla, jonka avulla selkeytetään ja hajautetaan ylläpitovastuuta. Tämä kuitenkin korjaa lähinnä seurauksia, eikä syitä seurausten takana. Kokonaisvaltaisempi ratkaisu ongelmaan olisi se, että laskentapohja ja mitoitusohje pystyttäisiin jollain tasolla integroimaan, jolloin päivitys myös laskentapohjaan tulisi ikään kuin ”automaattisesti” osana ohjeen päivitystä. Tämä vaatisi kuitenkin niin suuria panostuksia ohjeistuksen ja laskentapohjien kehitykseen, ettei niitä voida tehdä tämän työn puitteissa.



***Kuva 5.5** – Päällystysasemien laskentapohjan toiminnan periaatekaavio*

Päällystysasemien laskentapohjan toiminnan periaatekuvaaja on nähtävillä *kuvassa 5.5*. Päällystysaseman laskentapohja tukee kaikista laskentapohjista heikoimmin modulaarista lähestymistapaa. Siinä syöttöarvot on hajautettu osaksi jokaista alilaskentaa, pois lukien muutamat yhteiset muuttujat, joita hyödynnetään kaikissa osalaskennoissa. Tämä tekee input-rajapinnasta tilkkutäkkimäisen, joka on tulevan ylläpidon kannalta haasteellista. Toisaalta haasteen moduloinnille tuo myös päällystysasemien käyttöpiste-järjestyksen tekeminen. Käyttöpisteiden positiointi on perinteisesti ollut puhtaasti manuaalisesti tehtävää työtä, koska telojen järjestys päällystysasemilla on lähes jokaisessa projektissa erilainen, johtuen konseptien ja asiakkaan vaatimuksista. Tämä aiheuttaa moduloinnin vaatimalle yksinkertaiselle ohjelmarajapinnalle haasteita.

Myös osalaskennat itsessään ovat hankalasti päivitettävässä muodossa, koska niiden rakenne on jokseenkin orgaaninen, eivätkä ne seuraa mitään yhteistä logiikkaa. Toisaalta yhteisen logiikan löytäminen on myös haastavaa, koska päällystysaseman laskenta joutuu ottamaan kantaa laajaan kirjoon erilaisia mitoituselementtejä. Järvenpäässä tehtyjen haastattelujen pohjalta on myös alustavasti puhuttu päällystysasemien laskentapohjan kokonaisvaltaisemmasta uudelleen tekemisestä, mutta toistaiseksi resursseja tämän toteuttamiseen ei ole järjestetty. Mikäli päällystysaseman laskentapohjan kehitystyötä aletaan tekemään, niin jo alusta asti kannattaa ottaa huomioon tässä työssä esille tuotuja asioita ja miettiä laskentapohjan arkkitehtuuri sellaiseksi, että se on helppo integroida osaksi suurempaa systeemiä.



Kuva 5.6 *– Päällystysasemien laskentapohja*

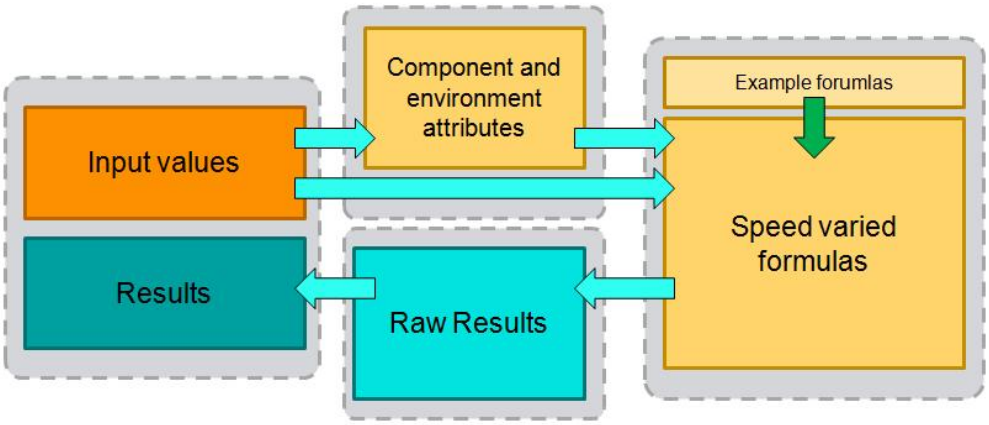
Kuvassa 5.6 on nähtävillä kuinka syöttösolut (kuvassa vaalean keltaisella) uppoutuvat osaksi koko laskentaa. Tämä logiikka jatkuu läpi koko laskentapohjan, mikä sopii sinänsä kohtalaisen hyvin laskentaan, joka suoritetaan vain kyseiselle rakenneryhmälle, koska tulokset muodostuvat nähtäville jokseenkin samaa tahtia kuin muuttujia täydentää. Koko linjan kattava laskentajärjestelmä asettaa kuitenkin vaatimuksia joihin nykyinen laskentapohja ei mukaudu joustavasti.

Päällystysasemien laskentapohjan avainomaisuuksia (vanhassa kontekstissaan):

- Päällystysasemien telojen järjestyksen muodostaminen on haastavaa
- Laskenta on kokonaisuudessaan haastavaa - konsepteja on paljon ja ne ovat hyvin erilaisia
- Laskennat ovat melko orgaanisia kokonaisuuksia, eikä kattavaa logiikkaa kaikkien laskenta-alueiden yli ole
- Syöttöarvot on hajautettu läpi koko laskennan
- Myös päällystysasemien laskennalla lasketaan (jälki)kuivatusryhmiä – jatkoa varten pitää tutkia mahdollisuus kuivatuslaskentojen integrointiin

5.3. Kalanteri

Kalanterilaskennan pohja liittyy erääseen toiseen kalanterilaskennan ohjelmistoon – CADI:in. Tämän projektin kannalta onnekkaasti myös CADI hyödyntää samanlaisille Excel-platformeille rakennettuja laskentoja. Myös CADI hyödyntää input- ja output-rajapintoja ja modulaarista laskentaa, joten kalanterin osalta integraatio linjalaajuiseen laskentaan oli helppo rakentaa. Samalla kalanterin laskentapohja toimii käytännön esimerkkinä siitä kuinka hyvin modulaarinen toteutustapa tukee systeemien rakentamista suuremmassakin mittakaavassa. Nyt pienellä työllä saatiin aikaan pohja, jota kaksi täysin erillistä järjestelmää voivat hyödyntää. Näin jos päivityksiä laskentaan tulee, niin ne tulee hoidettua kerralla kahteen eri järjestelmään.



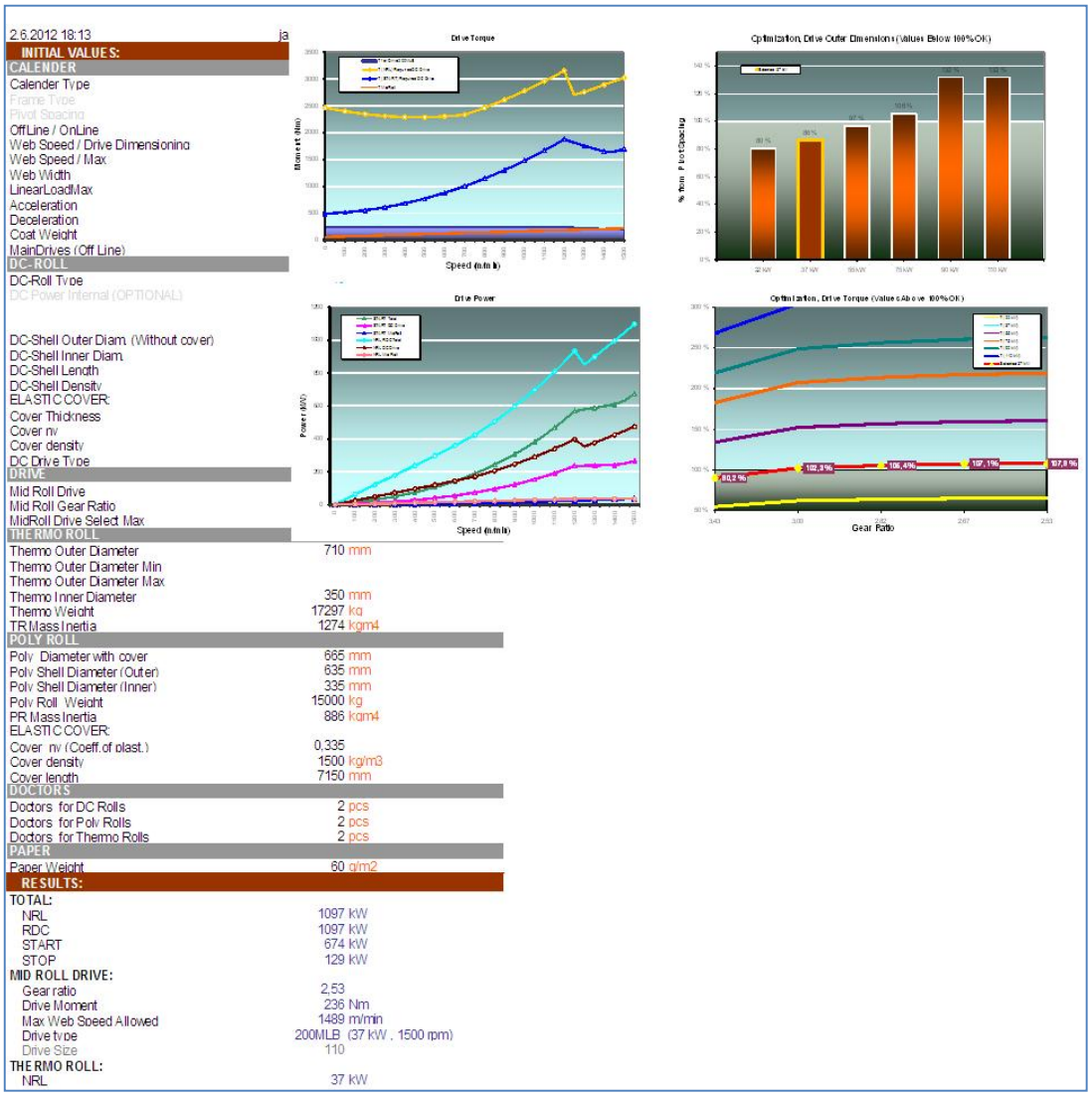
***Kuva 5.7** – Moninippikalanterin laskentapohjan toiminnan periaatekaavio*

Kuvassa 5.7 on esitetty kalanterin laskentapohjan toiminnan periaatekaavio. Kaaviosta on helppo huomata, että laskentapohjan suunnittelussa on otettu hyvin huomioon modulaarisen rakenteen vaatimukset – pohjaan on tehty helppoon pakettiin, yhteen paikkaan laskennan vaatimat inputit ja heti perään outputit. Kentät oli myös valmiiksi formalisoitu, jolloin rajapinnan rakentaminen tähän laskentaan oli jokseenkin triviaalia. Ainoastaan output-rajapinta piti rakentaa erikseen Tarmon vaatimaan formaattiin, jonka jälkeen laskentapohja oli yhteensopiva myös Tarmo 3:n kanssa.

Suurimman työn kalanterin laskentapohjan kanssa teettikin rajapinnan rakentamisen sijasta se, että se ei sisältänyt valmiiksi laskentaa ohjain- ja levitysteloille. Näille piti rakentaa kaavat ja implementoida ne osaksi laskentaa.

Toiminnaltaan moninippikalanteri on Tarmon vanhojen pohjien kaltainen, jossa laskenta on jaettu selkeisiin askeliin, jossa laskut referoituvat kerta toisensa jälkeen pidemmälle ja se sisältää vastaavat esimerkkikaavat, joiden avulla päivitettyjä kaavoja voi testata ilman, että joutuu koskemaan efektiivisiin kaavoihin.

Suurin ero kaikkiin muihin laskentapohjiin kalanterin pohjassa on se, että se ei ole nopeuden suhteen jatkuva, vaan diskreetti, jossa vastaus etsitään lähimmästä sopivasta nopeudesta. Tämä lähestymistapa juontaa juurensa CADI:in jonka toiminta vaatii sitä, että koneen nopeutta tutkitaan laajalla spektrillä, jotta kalanterin konsepti- ja laitevalinnat voidaan tehdä valitulla tavalla.



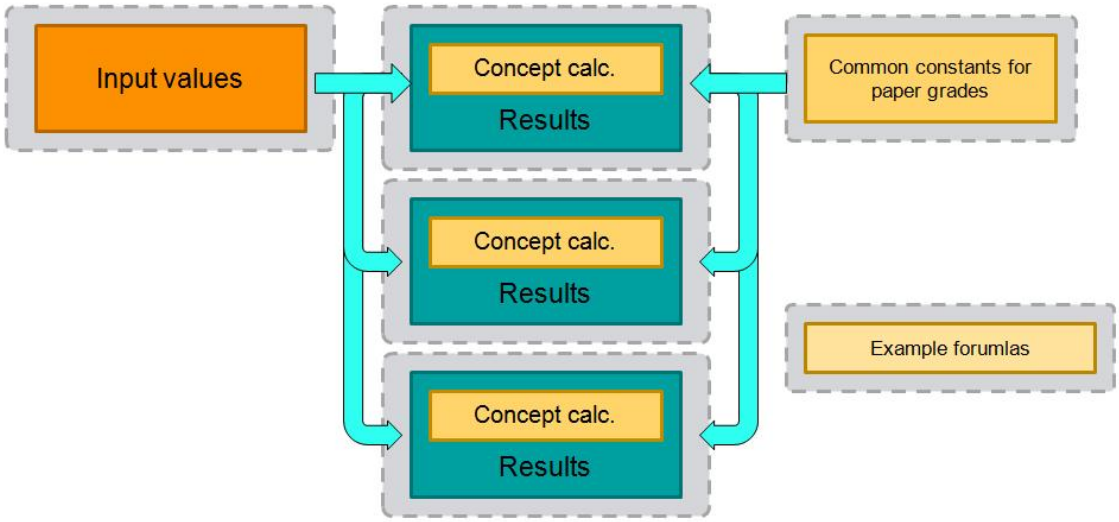
Kuva 5.8 *– Moninippikalanterin laskentapohja*

Kalanterin laskentapohjan avainomaisuuksia (vanhassa kontekstissaan):

- Laskentapohja noudattaa modulaarista lähestymistapaa jo valmiiksi, koska laskentapohja on sidottu myös toiseen järjestelmään
- Ei sisällä rakenneryhmän ohjaus- ja levitystelojen laskentaa
- Moninippikalanterin laskennassa suuressa osassa on nippien yli siirrettävä teho, koska kaikille käyttöpisteille ei saada suoraan tuotua riittävää käyttötehoa.

5.4. Rullain

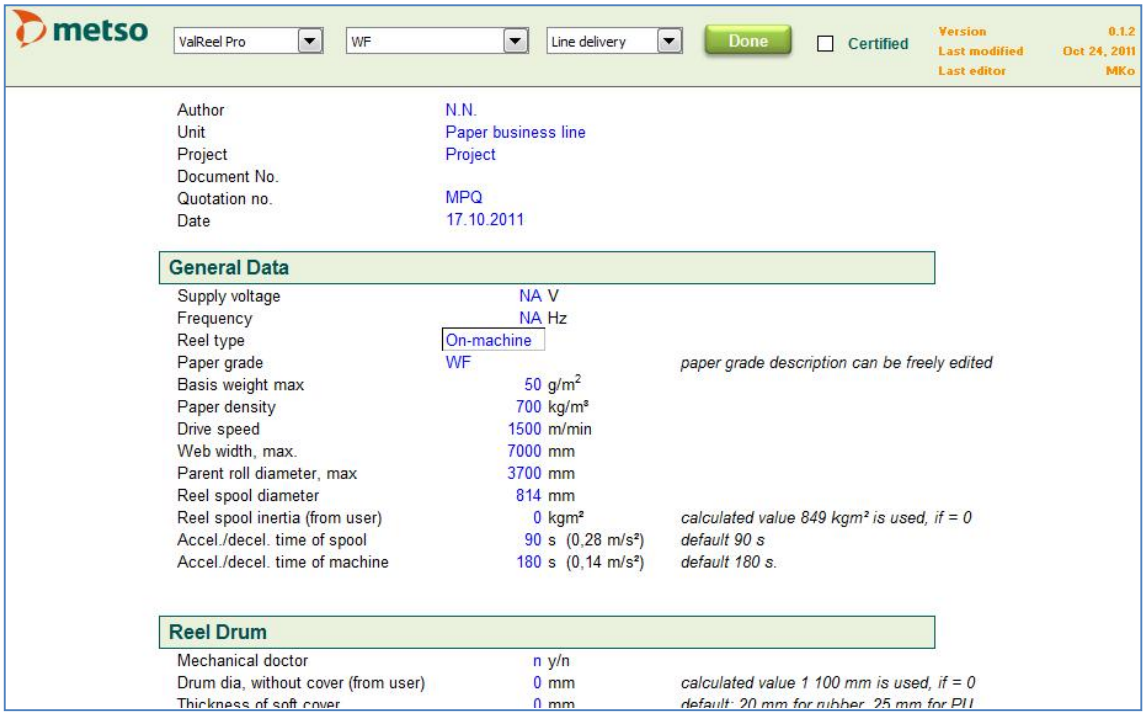
Rullaimen laskennan erityispiirteeksi laskennan integraatioprojektin kannalta voi todeta sen, että projektin läpiviennin aikana rullaimen laskentapohja tehtiin kokonaan uudelleen. Vanha laskentapohja oli sisäiseltä rakenteeltaan kömpelö ja laskentasisällöltään vanhanaikainen, joten päivitys oli odotettavissa. Vanhaan rullaimen laskentapohjaan oli projektin tiimoilta jo rakennettu rajapinta kokonaissysteemiin ja uuden pohjan myötä rajapinta piti rakentaa uudelleen. Uuden laskentapohjan tekijä oli kuitenkin perehdytetty kokonaissysteemin rakenteeseen ja näin ollen hän osasi laskentapohjan rakentamisessa varautua tähän. Päivitetyn laskentapohjan integrointi kokonaissysteemiin vastasi noin päivän työtä. Tämä antoi luottamusta kokonaisjärjestelmän hyvään ylläpidettävyyteen käytännön kokemusten kautta, sekä vahvisti modulaarisen rakenteen soveltuvuuden järjestelmään. Tässä käytännön esimerkissä laskentapohjan päivityksen teki ”kolmas taho” itsenäisesti ja integroinnin kokonaisjärjestelmään suoritti myynnin ylläpitoa vastaava taho. Tämä toimintamalli vastaa suunniteltua ylläpitorakenteen toimintaa (*kuva 2.7*).



***Kuva 5.9** – Rullaimen laskentapohjan toiminnan periaatekaavio*

Rullaimen uusi laskentapohja (*kuva 5.9*) on jaettu itsenäisiin rullainkonseptikokonaisuuksiin, eikä laskennoilla ole yhteisiä komponenttilaskentoja tai muitakaan keskinäisiä riippuvuuksia. Konseptilaskennat käyttävät yhteistä input-pöytää, sekä yhteistä

paperilajeittain toimivaa vakio-arvojen tietokantaa. Tämän lisäksi ohjelmassa on listattu erikseen laskennassa käytettävät kaavat, jotta niiden tarkistaminen olisi helppoa. Esimerkkikaavoja ei kuitenkaan sidota millään tavalla laskentaan tai kerrota missä päin laskentapohjaa kyseinen kaava on käytössä.



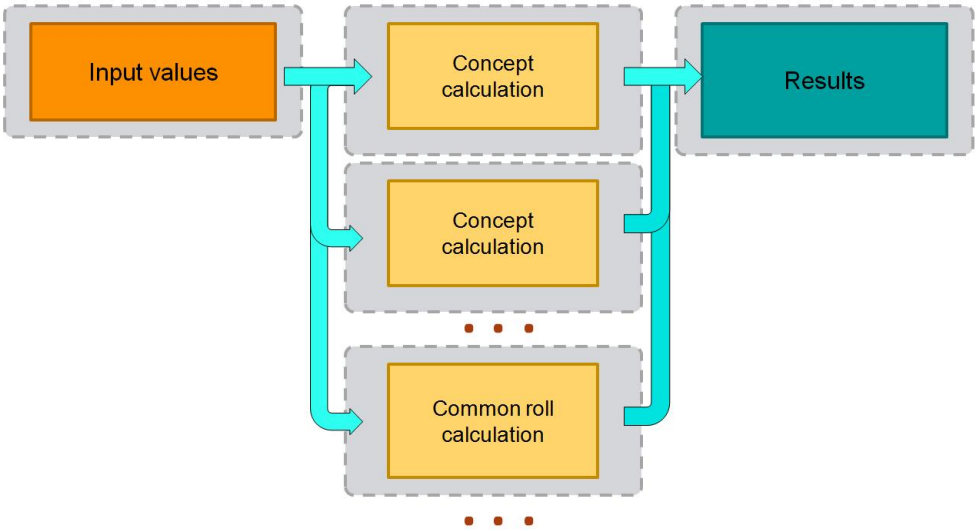
***Kuva 5.10** – Rullaimen laskentapohja*

Rullaimen laskentapohjan avainomaisuuksia (vanhassa kontekstissaan):

- Rullaimella ei ole erillistä mitoitusohjetta, kuten muilla rakenneryhmillä
- Laskenta ei pidä sisällään RDC-laskentaa
- Laskenta- ja tulosrakenne on jaettu rullainkonsepteittain
- Uusi laskentapohja muistuttaa rakenteeltaan pituusleikkuria

5.5. Pituusleikkuri

Pituusleikkurin laskentapohja muistuttaa pitkälti rullaimen laskentapohjaa. Tähän löytyy yksinkertainen selitys siitä, että rullaimen uuden laskentapohjan teki sama henkilö, joka on tehnyt myös pituusleikkurin laskentapohjan. Tämän ansiosta myös laskentapohjien rakenne (*kuva 5.11*) muistuttaa pitkälti toisiaan, joka puolestaan helpottaa ylläpidollisia tehtäviä.



***Kuva 5.11**– Leikkurin laskentapohjan toiminnan periaatekaavio*

Sekä rullain, että pituusleikkuri ovat tekijänsä ansiosta jo syntyjään vahvasti rajapintalähestymistapaan yhteensopivia. Tämä ei johdu niinkään siitä, että laskentapohjat olisivat alun perin olleet osa jotain isompaa laskentajärjestelmää, vaan siitä yksinkertaisesta syystä, että myös ihmiskäyttäjien näkökulmasta ”rajapintamaisesti” toteutettu UI on helposti lähestyttävä. Tässä tapauksessa ihmisille suunniteltu UI kääntyy helposti myös API:ksi, jolla laskentapohja voidaan sitoa osaksi integroitua linjalaajuista laskentajärjestelmää.

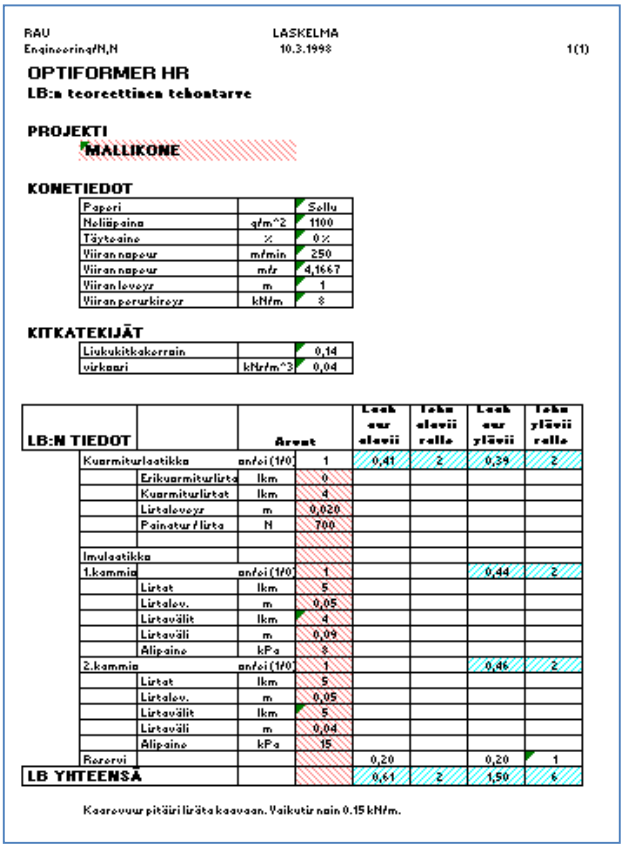
Merkittävin ero leikkurin laskentapohjan ja rullaimen välillä on se, että leikkurin laskennassa ei käytetä lainkaan yhteisiä paperilajikohtaisia muuttujia ja toisaalta myöskään esimerkkikaavoja ei esitetä ohjelmassa.

Pituusleikkurin laskentapohjan avainomaisuuksia (vanhassa kontekstissaan):

- Pituusleikkurilla ei ole erillistä mitoitusohjetta, kuten muilla rakenneryhmillä
- Laskenta ei pidä sisällään RDC-laskentaa
- Laskenta- ja tulosrakenne on jaettu pituusleikkurikonsepteittain
- Muistuttaa rakenteeltaan uutta rullainlaskentaa (laskentapohjan tekijä sama)
- Laskentaan ja syöttöarvoihin on rakennettu automaattinen imperiaalisen ja metrisen järjestelmän tunnistaminen

5.6. Suunnittelun erityistapausten laskupohja

Edellä esitetyt työkalut ovat niitä, joita myynnissä on käytetty tarjousinsinöörien toimesta, kun he tekevät alustavia tehontarvelistoja. Näiden lisäksi vastaavia tehontarvemäärittelyn työkaluja löytyy myös suunnittelun puolelta. Näitä työkaluja käytetään, mikäli halutaan varmistaa joitain erikoistapauksia sitovaa tehontarvelistaa varten.



RAU LASKELMA
Engineering/N,N 10.3.1998 1(1)

OPTIFORMER HR
LB:n teoreettinen tehontarve

PROJEKTI
MALLIKONE

KONETIEDOT

Paperi		Sellu
Neliöpaino	g/m^2	1100
Täyteaine	%	0 %
Viiran nopeus	m/min	250
Viiran nopeus	m/s	4,1667
Viiran leveys	m	1
Viiran perurkireys	kN/m	8

KITKATEKIJÄT

Liukukitkakerroin		0,14
virkaari	kN/m^3	0,04

LB:N TIEDOT			Arvot	Laak eur alavii	Tahk alavii rolle	Laak eur ylävii	Tahk ylävii rolle
Kuormiturlaatikka		onfoi (1/0)	1	0,41	2	0,39	2
	Erikuormiturlista	lkm	0				
	Kuormiturlistat	lkm	4				
	Listaleveys	m	0,020				
	Painatus/lista	N	700				
Imulaatikka							
1.kammio		onfoi (1/0)	1			0,44	2
	Listat	lkm	5				
	Listalev.	m	0,05				
	Listavälit	lkm	4				
	Listaväli	m	0,09				
	Alipaine	kPa	8				
2.kammio		onfoi (1/0)	1			0,46	2
	Listat	lkm	5				
	Listalev.	m	0,05				
	Listavälit	lkm	5				
	Listaväli	m	0,04				
	Alipaine	kPa	15				
Reservi				0,20		0,20	1
LB YHTEENSÄ				0,61	2	1,50	6

Kaarevuur pitäuri liräta kaavaan. Vaikutir nain 0.15 kN/m.

Kuva 5.12 *– Suunnittelun erityistapausten mitoituksen laskentapohja*

Kuvassa 5.12 on kuvankaappaus eräästä tällaisesta suunnittelun käyttämästä laskentatyökalusta. Näiden työkalujen laskentaperiaatteet ovat lähtökohtaisesti hyvin samanlaiset kuin aiemmin esitellyissä laskennoissa ja alun perin kumpikin laskentamalli onkin käyttänyt vastaavia kaavoja. Ajan myötä laskentapohjat ovat kuitenkin ajautuneet jokseenkin erilaisiksi, johtuen pääasiassa ylläpidon haasteista ja resurssien puutteesta. Laskennat ovat kuitenkin alun alkaenkin eronneet toisistaan. Tarjousvaiheen projekteissa täydellisen tarkan laskennan tuottaminen olisi liian raskasta ja hidasta sen vaiheen tarpeisiin. Tästä syystä tarjousvaiheen laskenta on muodostunut yksinkertaisemmaksi, jossa oletetaan monia asioita, jotta laskenta saadaan pysymään riittävän kevyenä. Suunnittelun työkaluissa puolestaan varianttien määrä on pidetty suurena, jotta laskennalla voidaan kattaa mahdollisimman laaja kirjo erikoistapauksia.

Tuntuu kuitenkin luontevalta ajatella, että kahta samankaltaista laskentaa on ”turha” ylläpitää kahdessa paikassa. Tästä syystä onkin syytä pohtia laskentojen tulevan kehityksen kannalta vaihtoehtoa, jossa suunnittelun nykyiset ”laajat” laskentapohjat yhdistettäisiin aiemmin esiteltyihin ”kevyempiin” laskentapohjiin. Kokonaisjärjestelmän modulaarinen, rajapintoja hyödyntävä lähestymistapa antaa tähän hyvän mahdollisuuden. Esimerkiksi laskentapohjat voidaan yhdistää yhdenmukaisiksi ja rakentaa tarjousinsinööreille ja laskentaeksperteille erilaiset käyttöliittymät. Tarjousinsinöörien käyttöliittymä voitaisiin rakentaa nykyisen tavan mukaisesti niin, että jokaista muuttujaa ei tarvitse syöttää, vaan asioita oletetaan valmiiksi. Laskentaeksperttien käyttöliittymä

voitaisiin puolestaan rakentaa niin, että he pääsevät tarvittaessa käsiksi kaikkiin tarvittaviin muuttujiin.

Nyt esitelty tarkempien laskentapohjien integraatio osaksi uutta järjestelmää vaatii kuitenkin niin paljon lisäselvityksiä ja työtä, että se rajataan tämän diplomityön ulkopuolelle. Alustavia keskusteluja laskentapohjien yhdistämisestä on kuitenkin käyty ja suunnittelun laskentaekspertit ovat osoittaneet alustavaa kiinnostusta integraation läpivientiin.

5.7. Laskentapohjien vertailu

Kuten edellisissä kappaleissa esitetyistä laskentapohjien periaatekaavioista voi helposti huomata, ovat eri rakenneryhmien laskentapohjat hyvin erilaisia, joka vahvistaa aiempaa päätelmää siitä, että modulaarisuus on oikea tapa lähestyä laskentapohjien integraatiota. Modulaarisuuden ja sen vaatimien rajapintojen turvin voidaan hyvinkin erilaiset laskentapohjat integroida samaan järjestelmään siten, että laskentapohjien erilaisuus ei välity käyttäjälle millään tavalla. Tällöin käyttäjä saa yhdenmukaisen käyttökokemuksen, joka on helpompi omaksua verrattuna siihen, että tämä joutuisi omaksumaan puolen tusinaa erilaista laskentapohjaa ja niiden käyttölogiikkaa.

Erilaisista lähtökohdista rakennetut laskentapohjat antavat arvokasta tietoa, jota voidaan hyväksikäyttää laskentapohjien tulevan edelleen kehittämisen kannalta. Pääohjelman arkkitehtuuria, modulaarisuutta ja rajapintoja tutkittaessa oli mielekästä analysoida myös laskentapohjien vahvuuksia ja heikkouksia suhteessa toisiinsa. *Kuvassa 5.13* on esitetty tämän analyysin tulos. On tärkeää huomata, että esitetty arvio on tämän diplomityön tekijän subjektiivinen näkemys (mutta laskentojen aiemmasta kehitystyöstä ulkopuolisena arvio on verrattain objektiivinen) eri laskentapohjien ja eri osa-alueiden toteutuksen tasosta arvioituna kvantitatiivisesti asteikolla 1-10. Laajempaa arviokyselyä toteutusten laadukkuudesta on mahdoton tehdä, koska kukaan muu ei ole joutunut tutustumaan kaikkiin laskentapohjiin yhtä perinpohjaisesti kuin tämän työn puitteissa on pitänyt tutustua. Kokonaistulosten suhteuttamiseksi eri osa-alueille on määritelty painotuskerroin, jolla kunkin osa-alueen tärkeyttä kuvataan.

Esitetyn arvion kokonaistuloksen perusteella Tarmon lähestymistapa osoittautui vertailluista pohjista parhaaksi. Toisaalta Tarmolla on laskentapohjan logiikan suhteen pieni etumatka, koska Tarmo on jo integroinut kolmen rakenneryhmän laskentapohjat itseensä. Näin ollen Tarmon suhteen on jo aiemmin tehty työtä, jolla laskentapohjien rakennetta on pystytty yhtenäistämään ja etsimään laskentojen kesken parhaita elementtejä. Heikoiten vertailusta suoriutui päällystysasema, jonka laskentapohjan päivitykseen on koettu sisäistä painetta, eikä tämän vertailun perusteella syyttä. Kolme muuta laskentapohjaa (kalanteri, rullain ja pituusleikkuri) sijoittuivat tasaisin pistevälein hieman Tarmon perään, niin että jokaisella oli hieman erilaisia heikkouksia ja vahvuuksia.

Laskentapohjien vertailu

Arviointi (1-10)

	Painotus	Päällystysasema	Kalanteri	Rullain	Pituusleikkuri	Tarmo
Käytettävyys						
Inputtien selkeys	8	6	9	8	8	9
Dynaaminen rakenne	4	2	3	5	5	9
Visuaalinen ulkoasu	4	2	8	6	6	9
Tiedonsyöttämisen helppous	8	6	8	9	9	9
Ohjelman läpinäkyvyys	7	4	4	8	8	4
Automatisointiaste	8	4	8	5	5	8
Käyttäjän ohjaaminen	7	3	4	7	7	8
Omaksuttavuus	7	5	5	7	7	8
Laskentapohjan toiminta						
Laskennan selkeys	6	7	9	7	8	6
Kaavojen selkeä ja koottu päivitysmahdollisuus	7	2	8	5	5	6
Muuta						
Dokumentointi	8	3	4	6	8	6
Painotetut pisteet		308	477	499	521	546

***Kuva 5.13** – Laskentapohjien kvantitatiivinen vertailu*

Tarmon laskentapohjat:

+Laskennan rakenne hyvin paloiteltu loogisiin kokonaisuuksiin

+Käyttöliittymäajattelu olemassa jo valmiiksi, joka tekee tiedon syöttämisestä intuitiivisempaa

+Laskentaa on automatisoitu ja käyttäjälle tarjotaan useissa asioissa oletusarvoja, joita tämä voi halutessaan käyttää

-Laskennan kulku itsessään vaatii jonkin verran perehtymistä, jotta siihen pääsee sisään

-Laskennan dokumentointi jättää toivomisen varaa

Päällystysaseman laskentapohja:

+Laskenta jaettu konseptin mukaisiin kokonaisuuksiin

-Laskenta melko hajanaisesti tehty

-Automatisointia ei ole nimeksikään

-Dokumentointi on heikkoa

-Ohjelman käyttö haastavaa

Kalanterin laskentapohja:

+Modulaarisuutta tukeva rakenne jo valmiiksi

+Sisältää graafisia esityksiä, jotka ovat käyttäjän kannalta ehdottomasti plussaa

+Laskennan rakenne hyvin palasteltu

+Automatisointi melko pitkälle viety

-Laskennan läpinäkyvyys jättää toivomisen varaa – yksittäisen laskentatuloksen muodostumisen seuraaminen on haastavaa
-Raskas rakenne isojen päivitysten kannalta

Rullaimen laskentapohja:
+Helppo ja koottu tietojen syöttäminen
+Laskentapohja on helppo omaksua ja käyttää
+Ohjelman toiminta ja sisältö on läpinäkyvää
-Rakenne on melko jäykkä isoja päivityksiä ajatellen
-Automatisointia ei ole viety erityisen pitkälle, mutta yksinkertaisen pohjan takia se ei ole välttämätöntä

Pituusleikkurin laskentapohja:
+Muutosten dokumentointi hoidettu hyvin
+/- Muutoin vastaavat plussat ja miinukset kuin rullaimen laskentapohjassa

5.8. Parhaat menetelmät

Edellisen vertailun pohjalta voidaan tutkia parhaita menetelmiä tiettyjen toimintojen ja ominaisuuksien toteuttamiseksi. Täydellistä laskupohjaa ei voida kuitenkaan saavuttaa ottamalla edellisen kappaleen vertailuista kunkin osa-alueen parhaita laskentapohjia ja kopioida niistä tietyn osa-alueen rakenne. Tämä johtuu siitä, että monet ominaisuudet ovat sidoksissa toisiinsa, jolloin muuttamalla jotain tiettyä ohjelman ominaisuutta, muutat myös jotain toista. Tämän takia parhaan menetelmän valinta ei ole triviaalia ja toisaalta paras menetelmä saattaa olla hieman erilainen kullekin eri rakenneryhmälle, johtuen eri rakenneryhmien erilaisista vaatimuksista laskennan suhteen.

Joitain universaaleja parannuksia voidaan kuitenkin löytää, jotka sopivat kaikkiin laskentapohjiin. Niitä ovat muun muassa seuraavat:

- Hyvin toteutettu dokumentointi niin kaavojen, koodin kuin muutostenkin osalta
- Input-rajapinnan selkeys
- Output-rajapinnan selkeys
- Visuaalisesti tarkoituksenmukainen ja miellyttävä ulkoasu

Hyvin toteutetussa *dokumentoinnissa* tulee kiinnittää huomiota siihen, että laskentaa on kommentoitu riittävästi, jotta ”ulkopuolisen” on helppo ymmärtää mitä laskennassa tapahtuu. Tämä edistää kokonaisuuden läpinäkyvyyttä ja omaksuttavuutta, sekä helpottaa ylläpitovastuun siirtämistä eteenpäin, joka karsii laskentapohjan elinkaarikustannuksia. Yhtälailla myös mahdollisesti käytetty koodi on hyvä kommentoida kattavasti. Tämä asia oli ennen tätä diplomityötä jäänyt lähes kokonaan tekemättä. Myös selkeä ja koottu muutospäivitys-logi on helppo ja tehokas tapa ylläpitää tietoa siitä mitä ohjelmasta on korjattu ja muutettu. Se antaa käyttäjille epäsuorasti luottamusta

myös ohjelmaan, sekä ylläpitoon, kun he voivat konkreettisesti halutessaan nähdä, että heidän esille tuomiin ongelmiin on myös käytännössä puututtu.

Input- ja output-rajapintojen selkeys on asia johon kokonaisarkkitehtuurin muutos tekee jo valmiiksi edistystä. Kokonaisarkkitehtuurin ansiosta kaikilla laskentapohjilla on määrämuotoiset rajapinnat, joiden läpi tietoa syötetään ja kaikkien laskentapohjien tieto on syötettävissä yhdenmukaisen käyttöliittymän läpi. Jos yksittäistä laskentapohjaa halutaan käyttää jostain syystä itsenäisesti, niin tällöin syöttöarvojen asetteluun ja käytettävyyteen kannattaa kiinnittää huomiota. Tavoitteena on kuitenkin, että laskentapohjia käytettäisiin pääasiassa pääohjelman kautta, jolloin yksittäisen laskupohjan käytettävyyden hiominen ei ole korkealle priorisoitava asia.

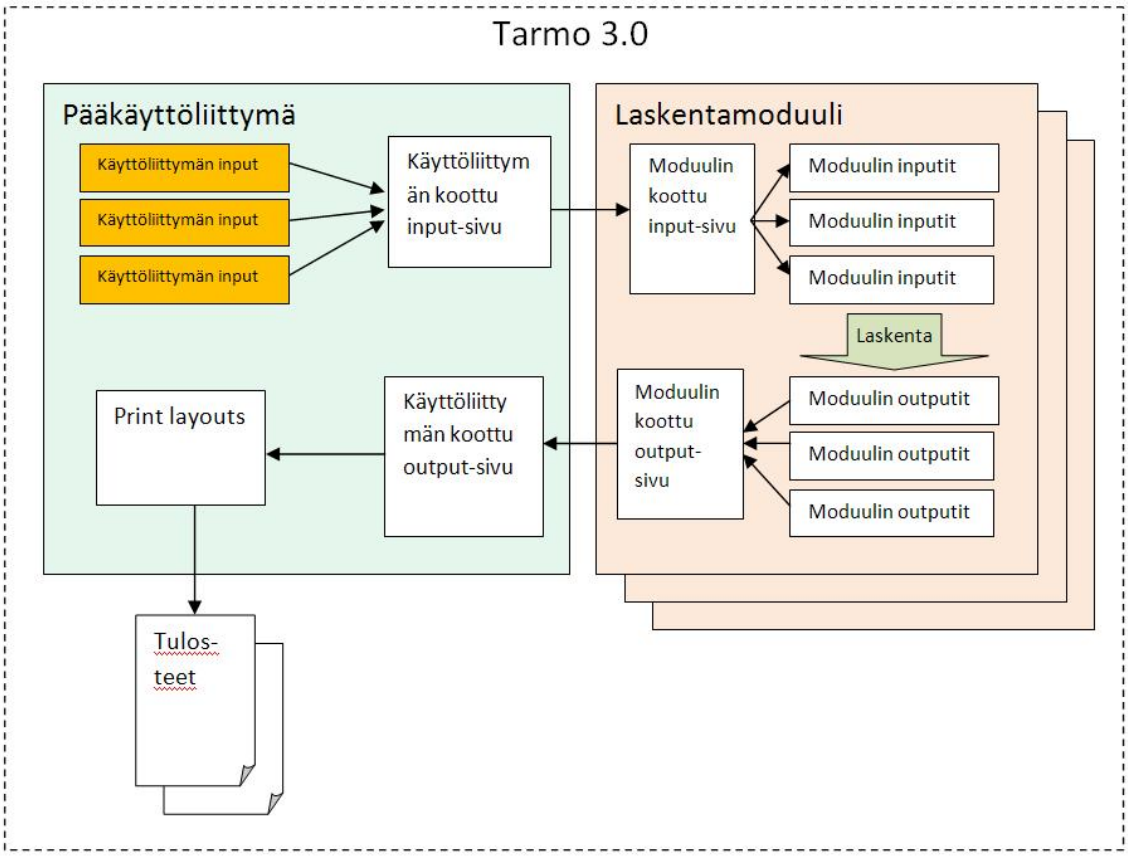
Visuaaliseen ulkoasuun tulee myös integraalinen parannus kokonaisjärjestelmän kautta, kun kaikkia laskentapohjia voidaan käyttää yhdenmukaisen käyttöliittymän kautta. Visuaaliseen ja tarkoituksenmukaiseen ulkoasuun voi kuitenkin kiinnittää huomiota, kun laskentapohjaa kehitetään edelleen, koska se usein parantaa asian ymmärrettävyyttä ja lähestyttävyyttä. Tämä voi osaltaan helpottaa laskentapohjan ylläpito- ja kehitystyötä.

6. KEHITETTY OHJELMISTO - TARMO 3.0

Tässä työssä aiemmin esiteltyjen periaatteiden mukaisesti on osana diplomityötä tehty Tarmo 3.0. Tämä ohjelma integroi alleen koko paperikoneen laajuuden: viiraosan, puristimen, kuivaimen, päällystysasemat, kalanterit, pituusleikkurit ja rullaimet. Uusi päivitys toi mukanaan uuden arkkitehtuurin, joka käyttää hyväkseen modulaarista rakennetta ja määriteltyjä rajapintoja. Tämän lisäksi päivitys implementoi aiemmin työssä käsiteltyjä parhaita menetelmiä osittain käytäntöön.

6.1. Ohjelman arkkitehtuuri

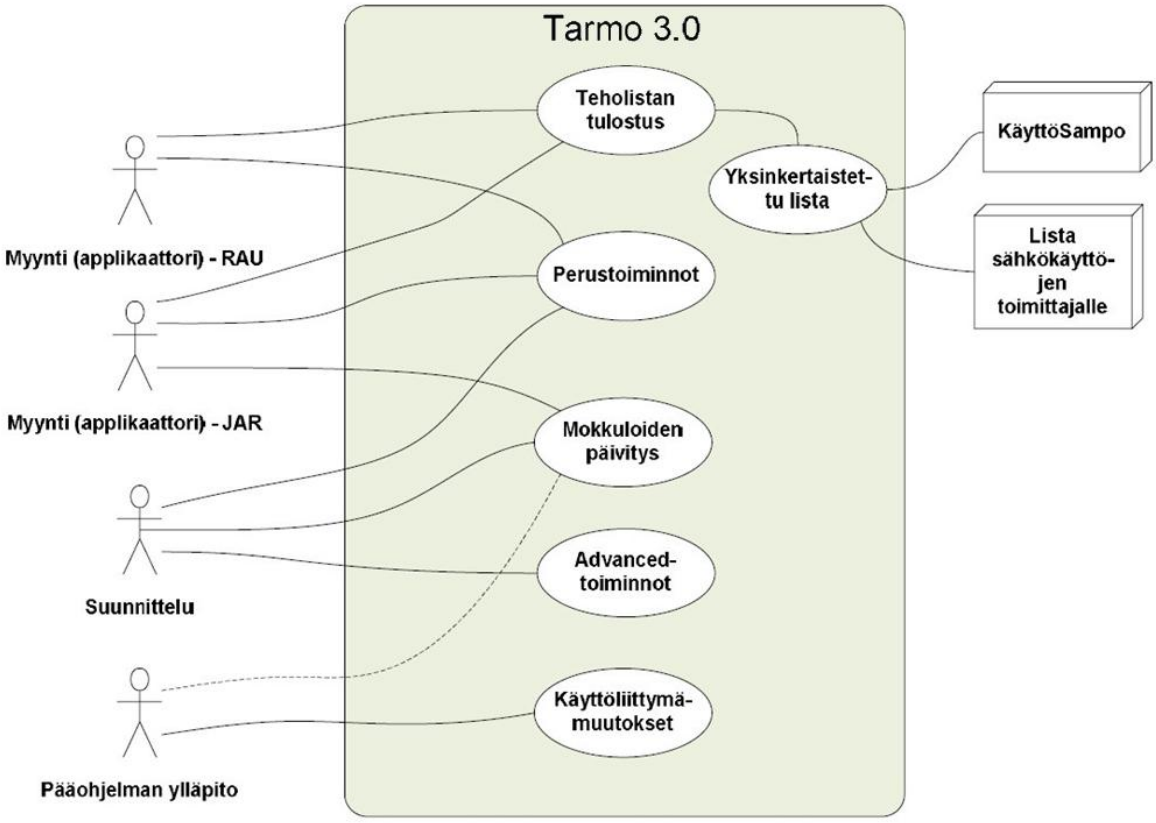
Ohjelman karkean tason arkkitehtuuri mukailee *kuvan 6.1* rakennetta. Tästä nähdään kuinka eri moduulit sitoutuvat toisiinsa ja kuinka kokonaisuus jakautuu selkeästi kahteen osuuteen: pääkäyttöliittymään ja laskentamoduuleihin.



***Kuva 6.1** – Tarmo 3.0:n karkean tason arkkitehtuuri*

6.1.1. UML-kaaviot

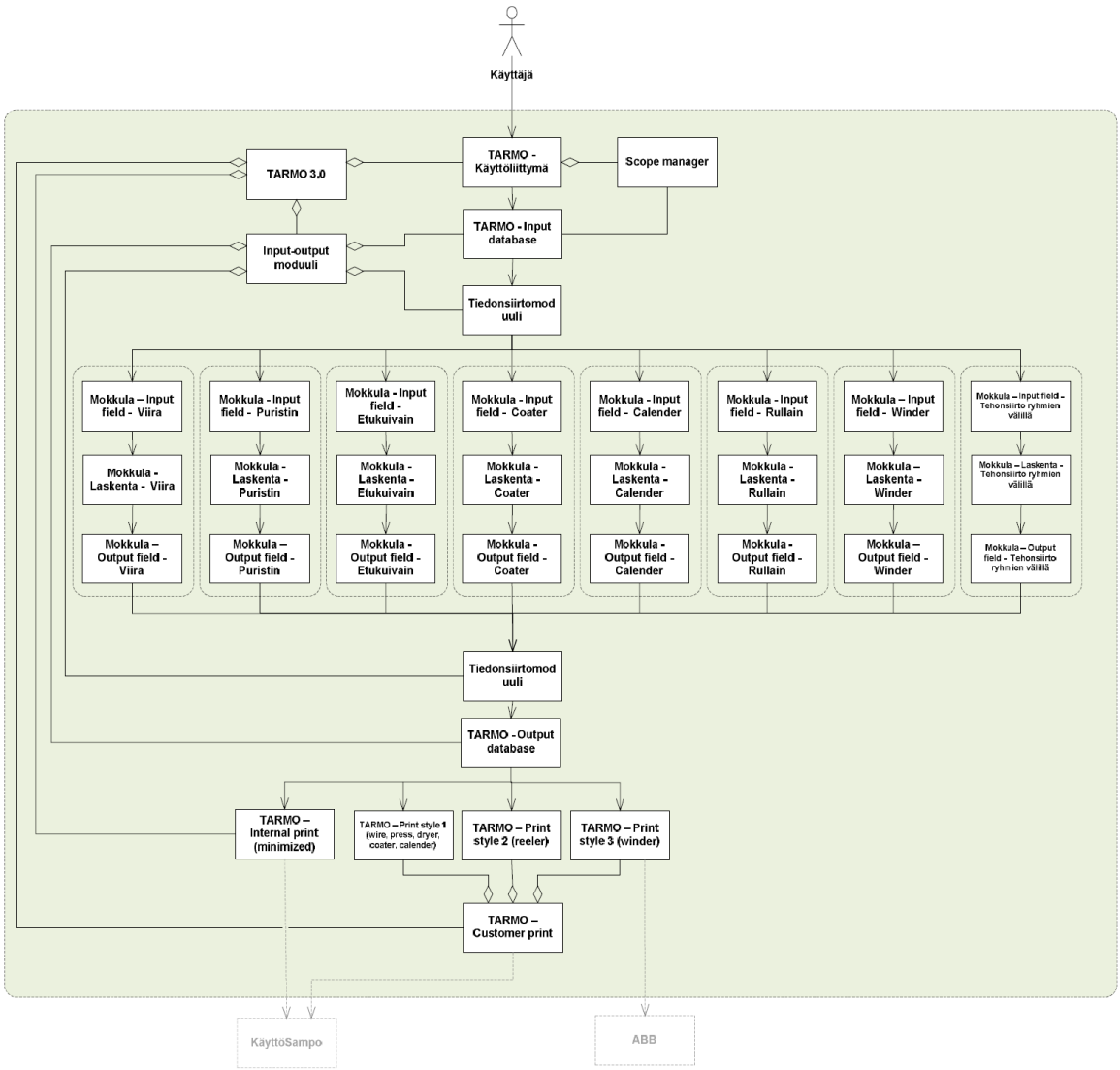
Tulevan ylläpidon helpottamiseksi Tarmo 3.0:n rakenteesta tehtiin UML-kaaviot (käyttötapauskaavio ja luokkakaavio), joiden avulla ohjelman rakenne on helpompi ymmärtää. Tällöin muutosten tekemisen vaikutuksia kokonaisuuteen on helpompi ymmärtää, jolloin muutosten tekemisen aiheuttamat kertautuvat virheet voidaan minimoida. Muutosten tekoa helpottaa myös kappaleessa 5 esitetyt rakennekuvaukset yksittäisistä laskentapohjista.



***Kuva 6.3** – Tarmo 3.0:n käyttötapauskaavio*

Käyttötapauskaaviossa (*kuva 6.3*) löydettiin viisi perustoimintoa, joita käyttäjäryhmien pitää pystyä tekemään: teholistan tulostus, perustoiminnot, kehittyneemmät toiminnot, moduulien päivitys, sekä käyttöliittymämuutokset. Käyttäjäryhmiä tunnistettiin neljä: myynnin applikaattorit, sekä Järvenpään, että Rautpohjan lokaatioissa, sekä suunnittelu, että pääohjelman ylläpito. Applikaattorien tulee pystyä käyttämään perustoimintoja, sekä tulostamaan teholistoja. Tämän lisäksi Järvenpään applikaattoreiden pitää tarvittaessa pystyä päivittämään laskentapohjia (moduuleita). Suunnittelun avainkäyttäjien on pystyttävä käyttämään perustoimintoja, kehittyneempiä toimintoja (esimerkiksi käyttää moduuleita itsenäisinä kokonaisuuksina) ja päivittämään laskentapohjia. Pääohjelman ylläpidon päätehtävä on käyttöliittymämuutokset, sekä tarvittaessa tukea laskentapohjien päivityksen ongelmissa tai toteuttaa pieniä päivityksiä itsenäisesti.

Käyttötapauskaaviosta huomataan myös ulkoiset järjestelmät, jotka voivat hyödyntää Tarmo 3:n lopputuloksia. Näistä toinen on Metson sisäinen järjestelmä (KäyttöSampo) ja toinen on alihankkijan järjestelmä (ABB:n sähkökäyttöjen valintaan käytettävä lista).



Kuva 6.3 *– Tarmo 3.0:n luokkakaavio*

Kuvassa 6.3 on nähtävissä karkean tason luokkakaavio Tarmo 3:n toiminnasta. Kaaviosta nähdään kuinka ohjelma rakentuu ja kuinka eri moduulit ovat sidoksissa toisiinsa. Laskentapohjien rakenteet käsitellään yksinkertaisuuden vuoksi tässä kaaviossa identtisinä, koska Tarmon kannalta laskentapohjien toiminta on toissijaista, kunhan input- ja output-rajapinnat ovat selvästi määriteltyjä. Tarkempi rakenne kustakin laskentapohjasta löytyy kappaleesta 5.

Kaaviosta nähdään että käyttöliittymä hyödyntää ”scope manager”:n avulla Tarmon input-tietokantaa, kun taas erilaiset tulosteet hyödyntävät Tarmon output-tietokantaa. Erilaisista tulostepohjista puolestaan koostetaan asiakastuloste käyttäjän

toiveiden mukaisella sisällöllä ja järjestyksessä. Tiedonsiirto-moduuli sen sijaan pitää huolen tiedon siirtämisestä input-tietokannasta laskentapohjiin ja laskentapohjista takaisin output-tietokantaan. Kuvassa on nähtävillä myös KäyttöSampon ja ABB:n sähkökäyttölistan hyödyntämät tulostepohjat.

6.2. Tarmo 3:n käyttöliittymä

Tarmo 3:een toteutettu käyttöliittymä on ulkoasultaan vahvasti sukua usealle myyntiosastolla käytetylle ohjelmistolle. Tällä pyritään siihen, että eri ohjelmien käyttökokemukset olisivat peruskäyttäjän näkökulmasta mahdollisimman samanlaisia. Tämä helpottaa uusien ohjelmien sisäistämistä, kun niiden toimintalogiikka on jo aiemmin tuttu, vaikkakin eri yhteydestä.

Tarmon käyttöliittymän perussuunnittelu nojaa vahvasti “Object-oriented analysis and design with the unified process” [Satzinger, Jackson, Burd 2005] kirjassa esitettyihin huomioihin käytettävyyden suunnittelusta.

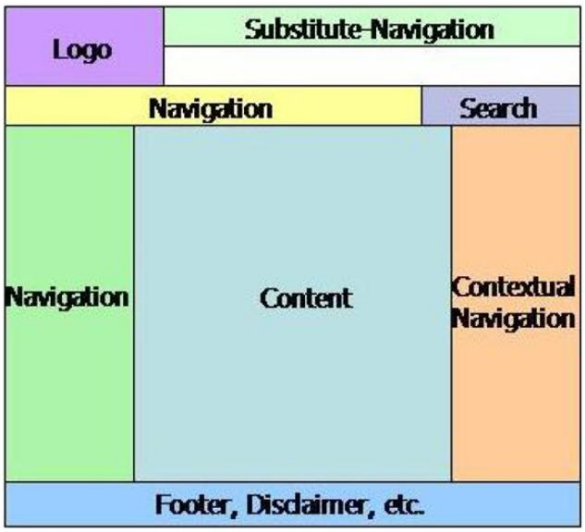
Käyttöliittymäsuunnittelun kahdeksan kultaista sääntöä:

- Pyri yhdenmukaisuuteen ja jatkuvuuteen.
- Mahdollista pikanäppäinten käyttö säännöllisille käyttäjille.
- Tarjoa informatiivista palautetta.
- Suunnittele dialogeja, jotka ohjaavat oikeaan suuntaan.
- Tarjoa yksinkertainen tapa käsitellä ongelmatilanteita.
- Salli helppo muutosten peruminen.
- Tue sisäistä hallintamahdollisuutta Support internal locus of control.
- Vähennä lyhytaikaisen muistin kuormitusta.

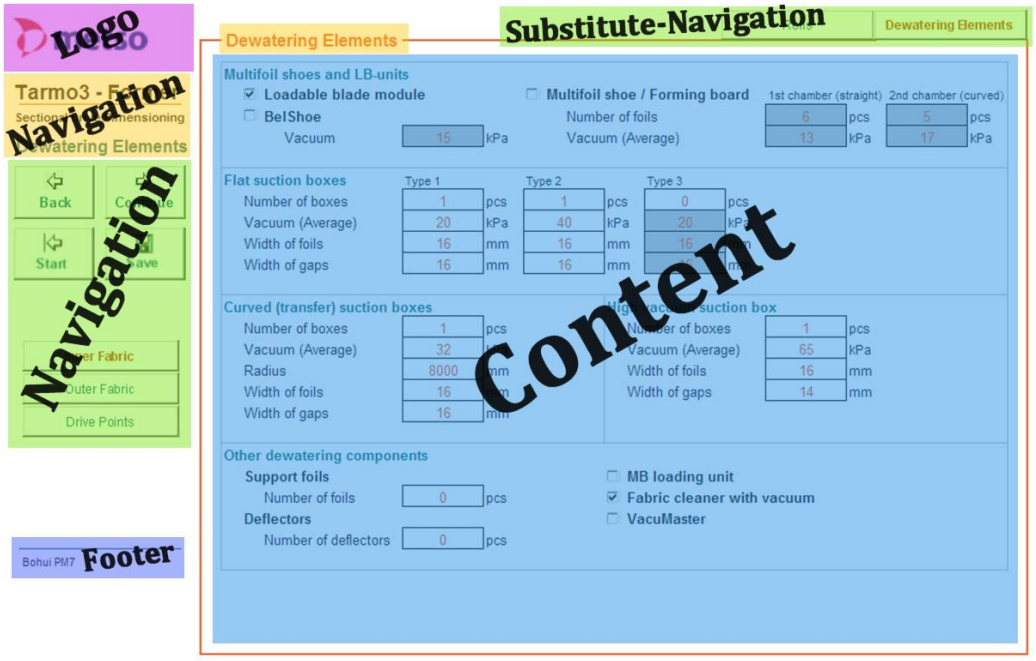
Käyttöliittymän ymmärtämiseen kirja [Satzinger, Jackson, Burd 2005] listaa kolme pääasiaa. Ensimmäinen on fysikaaliset aspektit, joka sisältää käyttöliittymän kosketeltavat asiat, eli näppäimistöt, hiiret, näytöt yms. Toinen kokonaisuus on havainnolliset asiat, joka sisältää kaiken mitä käyttäjä näkee, kuulee, tuntee jne. Tämä tarkoittaa käytännössä menetelmiä, joilla käyttöjärjestelmä kommunikoi käyttäjänsä kanssa. Kolmas kokonaisuus on konseptitason aspektit, jotka liittyvät käytettävään järjestelmään ja sen ymmärtämiseen. [Immonen 2010]

Tämän lisäksi kirjassa [Satzinger, Jackson, Burd 2005] korostetaan toimintojen näkyvyyttä. Käyttäjän tulee voida nähdä kaikki toiminnot halutessaan, mutta kuitenkin siten että ne tulevat näkyviin siten, että harvemmin käytettävät toiminnot eivät hankaloita normaalia käyttöä. Toimintojen tulee myös tarjota palautetta käyttäjälle, joka kertoo, että ohjelma tekee käyttäjän toivomaa asiaa. Myös käyttöliittymän nappien ja toimintojen tulisi kertoa toiminnallisuudestaan jo pelkän ulkoasunsa puolesta. Tämä nopeuttaa käyttöä ja tekee siitä intuitiivisempaa. [Immonen 2010] Laskennan läpinäkyvyyden lisäämisen tulee olla tärkeässä asemassa käyttöliittymän edelleen kehityksessä, koska

tällä hetkellä useat kokeneemmat käyttäjät toivoisivat, että laskennan kulkua pystyisi tarkkailemaan helpommin.



Kuva 6.4 *– Layout suunnitelma portaalille [Welie 2004]*



Kuva 6.5 *– Tarmo 3:n layout peruskäyttöliittymäsivulle*

Tarmo 3:ssa käytössä oleva käyttöliittymä on perusnäkymältään (*kuva 6.5*) hyvin samankaltainen kuin Weilien ”Patterns in Interaction Design”-artikkelissa esitetty portaalikäyttöliittymän mallipohja (*kuva 6.4*). Tarmo 3:n käyttöliittymästä löytyvät käytännössä kaikki samat elementit paitsi hakukenttä. Myös elementtien asettelu muistuttaa lähes identtisesti artikkelissa esitettyä mallipohjaa. Tämä käyttöliittymäpohja on todettu Metsossa käytössä toimivaksi, joten vastaavaa käyttöliittymäpohjaa on otettu käyttöön lukuisissa sovelluksissa. Tarmo 3:n suhteen käyttöliittymään on lisätty joitain käyttöä helpottavia yksityiskohtia, jotta käyttöliittymä olisi aiempaa intuitiivisempi.

Nykyisen Tarmon käyttöliittymän suunnittelussa on hyödynnetty ”The usability engineering lifecycle: A practitioner's handbook for user interface design” [Mayhew 1999] kirjassa esiin tuotuja asioita:

1. Hyödyntää automaation tarjoamia mahdollisuuksia.
2. Suunnitella työ siten, että se tukee tehokkaammin toiminnan tavoitteita.
3. Minimoida uudelleenkoulutuksen tarve, hyödyntämällä uudessa tuotteessa mahdollisimman paljon käyttäjien olemassa olevaa tietämystä tehtävistä sekä maksimoimalla suorituskyky (efficiency) ja tehokkuus (effectiveness), sovittamalla ihmisen kognitiiviset rajoitteet ja mahdollisuudet oikeaan tehtäväkontekstiin.

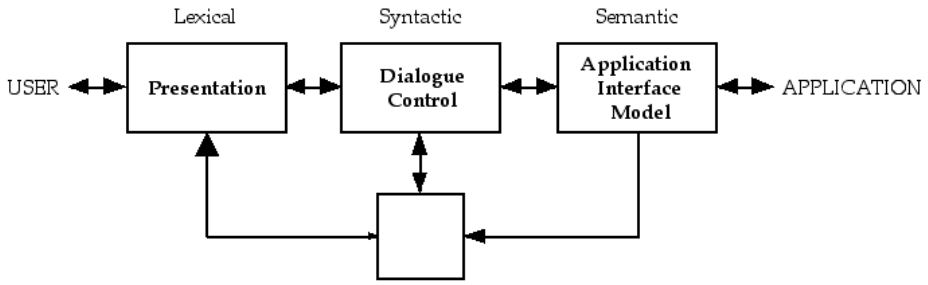
Tarmo *hyödyntää automaation* tuomia mahdollisuuksia tehokkaasti. Esimerkiksi käyttöliittymän dialogin eteneminen on automatisoitu siten, että käyttöliittymä mukautuu käyttäjän valintojen mukaan, niin että käyttäjälle näytetään tuhansien muuttujien joukosta ainoastaan ne joita käyttäjän tarvitsee nähdä. Tämän lisäksi käyttöliittymä pyrkii tukemaan käyttäjää muuttujien täyttämisessä tekemällä osasta muuttujista aiempien valintojen perusteella suuntaa antavia arvauksia. Automatisointia on kuitenkin mahdollista viedä pidemmälle etenkin sellaisilla tavoilla, jotka tukevat ohjelman käytön läpinäkyvyyttä – tästä puhutaan enemmän kappaleessa 6.2.1.

Jos mietitään käyttöliittymää ja sen suunnittelua, niin harvoin tulee ajatelleeksi, että sen suunnittelulla suunnittelee osittain myös sitä kuinka työ toteutetaan ja jopa sitä, kuinka hyvin *käyttöliittymä tukee toiminnan tavoitteita* suuremmassakin mittakaavassa. Tarmossa tätä ajattelua on tuotu mukaan käyttöliittymäsuunnitteluun. Myynti on organisaationa muuttunut yhä kokonaisvaltaisemmaksi linjatoiminnoksi, jota myös uuden Tarmon käyttöliittymä ja rakenne tukee: ohjelma ja sen käyttö tukee koko konelinjan laajuutta yhdenmukaisen käyttöliittymän kautta.

Tarmo 3:n käyttöliittymäsuunnittelussa on pyritty *minimoimaan uudelleenkoulutustarve*, jota uusi ohjelma vaatii. Vanhalla Tarmolla on suhteellisesti suurin käyttäjäryhmä, kun katsotaan moduuleita joita uusi Tarmo on yhdistänyt samaan kokonaisuuteen. Osittain tämän takia myös uuden Tarmon käyttöliittymä on suunniteltu pitkälti vanhan Tarmon mukaisesti, jolloin suurin käyttäjäryhmä tuntee jo valmiiksi Tarmo 3:n käyttöliittymän ja sen toimintalogiikan. Muiden käyttäjäryhmien omaksumisen helppoutta on pyritty ylläpitämään siten, että kyseisten moduulien käyttöliittymiin on pyritty implementoimaan vastaavaa toiminnallisuutta ja logiikkaa kuin alkuperäisissä laskentapohjissa oli. Tämä laskee uuden käyttöliittymän omaksumiskynnystä, koska käyttäjä voi huomata, että hän pystyy tekemään yhä samat asiat kuin aiemmin ja jopa helpommin.

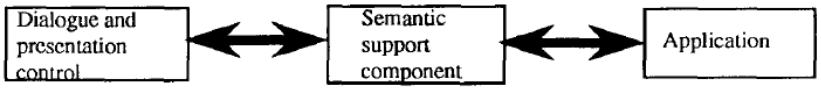
6.2.1. Käyttöliittymän kehittäminen eteenpäin

Vaikka nykyisen Tarmon käyttöliittymää on jo mietitty, hiottu ja käytetty hyvin kokemuksin, niin siinä on yhä aspekteja, joita voidaan parantaa.



***Kuva 6.6** – Seeheimin malli tietokone-ihmis-rajapinnasta*

Perinteinen malli ihmis-kone-rajapinnan kuvaamiseen on niin sanottu Seeheimin-malli (*kuva 6.6*). Tässä mallissa käyttöliittymä jaetaan kolmeen tasoon: esitystasoon, dialogin-hallintaan, sekä ohjelman kommunikaatiorajapintaan. Esitystaso tarkoittaa näkymää, jonka käyttäjä näkee ohjelmaa käyttäessään. Dialogin hallinta puolestaan hallitsee dialogin rakennetta, logiikkaa ja jatkumoa eri käyttöliittymänäkymien välillä. Kommunikaatiorajapinta puolestaan pitää huolen, että käyttäjän käyttöliittymään siirtymä informaatio siirtyy oikealla tavalla ohjelman tieto- ja ongelmanratkaisujärjestelmiin. Tätä tasoa voi tietyllä tapaa pitää käyttöliittymärakenteen semanttisena tasona, joka tulkitsee käyttäjän syötteiden merkityksiä, jotta ohjelma ”ymmärtäisi” käyttäjän toiveita. [Rathnam & Mannino 1995]



***Kuva 6.7** – Szekelyn käyttöliittymämalli*

Seeheimin-malli antaa kapean kanavan ohjelman ytimen ja käyttäjän välille. Tähän toisenlaisen lähestymistavan antaa Szekelyn-malli (*kuva 6.7*), joka pyrkii hämärtämään rajaa ohjelman ytimen ja käyttöliittymän välillä. Se ikään kuin yhdistää esityksen ja dialogin hallinnan yhdeksi kokonaisuudeksi, joka kommunikoi ohjelman ytimen kanssa semanttisen tuki-moduulin kautta. Tämä lähestymismalli antaa mahdollisuuden ”elävämpään” käyttöliittymään, joka keskustelee aktiivisesti ohjelman kanssa. Tämä tukee käyttökokemusta lisäämällä läpinäkyvyyttä ja hallinnan tunnetta. Erityisesti läpinäkyvyyden lisääminen on tärkeää, koska jo tällä hetkellä läpinäkyvyyden puute on koettu Tarmon kanssa haasteeksi. Szekelyn-mallin käyttö tukee myös modulaarisuutta [Rathnam & Mannino 1995], joka tekee kyseisen mallin käytöstä erityisen houkuttelevan vaihtoehdon, mikäli käyttöliittymän edelleen kehitystä halutaan lähteä systemaattisesti toteuttamaan.

7. YHTEENVETO JA JATKOKEHITYSIDEOITA

Työn päätavoitteena oli yhdenmukaistaa ja tehostaa paperikoneen käyttötehojen määrityksen prosessia. Tämän lisäksi toivottiin, että jatkossa käyttötehot voisi määrittää yhden henkilön toimesta, kun olemassa olevassa tilanteessa määritykseen tarvitaan pahimmillaan jopa viiden tarjousinsinöörin työpanosta. Tavoitteiden saavuttamiseksi työssä käytiin läpi seuraavat kokonaisuudet:

Aluksi määritettiin *linjakäyttöjen tehontarpeen suhde energian kulutusarvolaskentaan*. Tässä kappaleessa pyrittiin osoittamaan tehontarvelaskennan osuus suuremmasta kulutusarvojen määrityksen kuvasta. Kuten kappaleessa todettiin, on käyttöjen osuus sähkönkulutuksen kokonaismäärästä merkittävä. Näin ollen kyseessä on tärkeä kokonaisuus, jolloin ei voitu ottaa käyttöön liian suuria approksimaatioita prosessin tehostamisen nimissä. Toisaalta järjestelmän suunnittelussa piti ottaa huomioon myös mahdolliset tulevat toiveet suunnittelun rakenneryhmien omien tehonmääritysohjelmien yhdistämisestä linjatason laskentaan (toisin sanoen uuteen Tarmoon).

Tämän jälkeen selvitettiin *tarjousprosessin rakenne*, jotta ymmärrettäisiin miten tehonmääritys etenee suhteessa tarjousprosessiin myyntivaiheessa. Tarjousprosessin ymmärtäminen auttoi uuden järjestelmän vaatimusmäärittelyssä, jotta se pystyisi vastaamaan sille asetettuihin toiveisiin. Toisaalta tarjousprosessi määritti myös tapaa, jolla järjestelmän käyttöliittymä kannatti toteuttaa, jotta se tukisi tarjousprosessia oikealla tavalla.

Jotta linjatason laskennan integraatiota voitiin perustellusti suunnitella, tuli ymmärtää myös paperikonelinjan *koneryhmien tyypilliset piirteet käyttöjen kannalta*. Tässä kappaleessa perehdyttiin karkealla tasolla siihen millaisia rakenneryhmät ovat ja millaiset komponentit, sekä prosessin vaiheet aiheuttavat niissä tehon kulutusta.

Koska tavoitteena oli kehittää käyttökelpoinen järjestelmä, syvennyttiin seuraavassa kokonaisuudessa *systeemien kehityksen teoriaan*, jonka tuella pystyttiin määrittelemään ja toteuttamaan vaatimusten mukainen ja toimiva järjestelmä. Läpikäytyjen teorioiden perusteella parhaiksi vaihtoehdoiksi järjestelmän toteutukseen valikoituvat modulaarinen lähestymistapa, joka toteutettaisiin kehitysprosessilla, joka korostaa käytettävyyttä, sekä nopeita prototyyppejä.

Työn tuloksena rakennettiin integroitu järjestelmä, joka pystyi hyödyntämään vanhoja, toimivia laskentapohjia. Tämän toteutustavan valintaan oli kaksi pääsyytä: aiempia laskentapohjia on laajasti testattu ja niiden virheitä korjattu, jolloin pyörän keksiminen uudelleen ei ollut järkevää. Toisekseen vanhoille laskentapohjille löytyi myös niitä tuntevia ihmisiä, jotka mahdollistavat ylläpidon hajautuksen. Jos ylläpitoa ei saataisi hajautettua, niin näin suuren järjestelmän ylläpito rämettyisi nopeasti. Tästä syystä

seuraavaksi käsiteltiin *vanhojen rakenneryhmien laskentapohjien analysointi* ja niiden pääpiirteiden määritys. Tämän perusteella pystyttiin suunnittelemaan jokaiselle rakenneryhmän laskennalle kokonaisuuteen sopiva rajapinta ja räätälöinti, jolla ne voitiin saattaa osaksi integroitua järjestelmää.

Suunnitelman mukaisesti toteutettiin myös ohjelmisto (*kuva 7.1*) - *Tarmo 3.0*, joka yhdisti modulaarisen rakenteen avulla yhteen pakettiin koko linjan kattavan tehontarpeen määrityksen. Tämän lisäksi Tarmo 3.0:n arkkitehtuuri tukee sitä, että pienellä työllä voidaan rakentaa moduuleita, joilla ulkopuoliset ohjelmat tai järjestelmät voivat käyttää Tarmo3:a, sen tuloksia tai alimoduuleita hyväkseen.

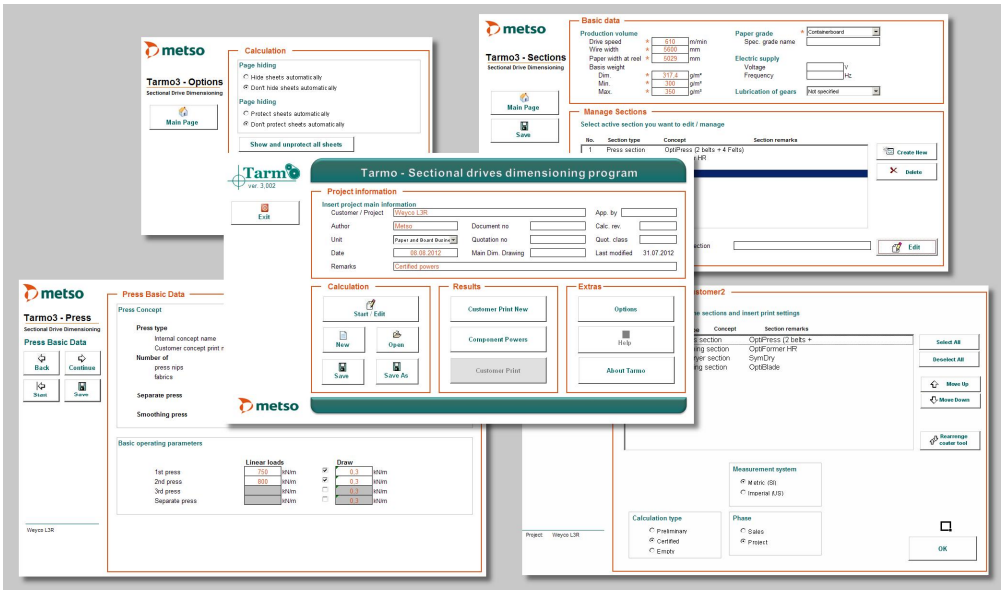
Integroitu tehontarvelaskennan ohjelmisto suunniteltiin siten, että se ratkaisisi tai ainakin lieventäisi työn alussa esiin tuotuja ongelmia ja tavoitteita:

- Pitkä läpimenoaika ja sen varianssi, sekä useiden henkilöiden työpanos → Suunniteltiin järjestelmä, jolla yksi ihminen voi tehdä laskelman, jolloin läpimenoaika ja varianssi pienenee → Helpommin hallittava kokonaisuus
- Tarjousvaiheen revisioinnin työläys → Järjestelmä mahdollistaa sen, että yksi ihminen voi hoitaa revisioinnin, eikä manuaalisia erillisten teholistan osien parsimisia tarvita, kun ohjelma tekee tämän automaattisesti
- Dokumenttien yhdenmukaisuusongelma → Laskennat on järjestelmän kautta pakotettu samaan muottiin, joten lopputuotokset ovat yhdenmukaisia
- Moninaiset toimintamallit → Järjestelmän ansiosta kaikki laskennat ovat samassa kokonaisuudessa ja viedään loppukäyttäjän näkökulmasta läpi samalla tavalla
- Ylläpitovastuu on keskittynyt liikaa yhdelle taholle → Uuden järjestelmän arkkitehtuuri mahdollistaa ylläpitovastuun hajauttamisen useammalle taholle, jolloin päivitykset tekevät sopivimmat mahdolliset henkilöt, jonka tulisi lyhentää päivitysten läpimenoaikoja
- Käyttöjen ylimitoituksen tarkistaminen → Kuivatusosan teholaskelmia muutettiin, siten että moottorit mitoitetaan lähemmäs optimaalista ajoaluetta
- Kilpailukyvyn lisääminen → Nopeampi ja ketterämpi linjakäyttöjen mitoitusprosessi

Vaikka työn puitteissa kehitetyn kokonaisarkkitehtuuri onkin uudenlainen, niin kokonaisuus on pyritty rakentamaan selkeäksi ja hyvin kommentoiduksi, joka helpottaa ylläpitoa. Toisaalta laskentamoduulien sisäinen arkkitehtuuri on vapaa, joten jokainen laskentapohjan ylläpitäjä voi valita haluamansa laskenta-arkkitehtuurin ja hyödyntää vanhoja laskentoja halutessaan täysmääräisesti - ohjelmiston API pitää huolen, että hyvinkin erinäköiset laskentapohjat keskustelevat pääohjelman kanssa halutulla tavalla.

Uuden ohjelmiston käyttöliittymä on rakennettu siten, että se tukee käyttäjää mahdollisimman paljon, koska käsiteltävät kokonaisuudet ovat suuria, joita kukaan käyttäjä ei pysty hallitsemaan kokonaisvaltaisesti. Käyttöliittymäsuunnittelu kunnioittaa myynnissä jo tehtyä kehitystyötä käyttöliittymien osalta. Näin ollen uuden Tarmon käyttöönotto myynnin järjestelmien vanhoille käyttäjille on helppo ja tietyllä tapaa huomaamaton prosessi. Järjestelmiin ennestään tutustumattomille muutos on niin ikään

suurempi, mutta he saavat käyttöönsä laajasti testatun ja hiotun käyttöliittymän, jonka omaksuminen on suhteellisen nopea prosessi.



***Kuva 7.1** – Kuvankaappauksia Tarmo 3.0:sta*

Alustavien testien perusteella ohjelmassa on yhä paljon asioita joita voidaan hioa paremmiksi, mutta kokonaisuudessaan ohjelma täyttää sille asetetut vaatimukset. Aika ja käytännön kokemukset tulevat osoittamaan kuinka nopeasti ohjelma omaksutaan laajaan käyttöön ja kuinka paljon se loppujen lopuksi tehostaa käyttötehojen määrityksen prosessia. Sivutuloksina työstä on syntynyt materiaalia, jota on pystytty hyödyntämään useassa muussa asiantuntijaohjelmassa Metson sisällä.

7.1. Jatkokehitysideoita

Työn edetessä tuli eteen paljon asioita, joita ei voitu tämän työn puitteissa toteuttaa. Seuraavassa on käsitelty näitä jatkokehitysideoita lyhyesti.

Laskentapohjien *kaavoja olisi mahdollista yhdenmukaistaa pidemmälle*. Esimerkiksi kaapimet, huovanjohtotelat, paperinjohtotelat (ym.) voitaisiin upottaa yhteiseen laskentaan. Kaavojen ylläpito helpottuisi yhteisten komponenttilaskentojen myötä, kun kaavapäivitykset periytyisivät automaattisesti kaikkiin laskentoihin. Nykyisellä laskenta-arkkitehtuurilla kukin rakenneryhmä laskee komponenttien tehonkulutukset erikseen.

Muutamissa työn aikana pidetyissä palavereissa on ollut esillä ajatus siitä, että *Tarmo voitaisiin yhdistää suunnittelun omien laskentapohjien kanssa*. Tämä vähentäisi päällekkäisyyksiä, helpottaisi päivityksiä ja laajentaisi ohjelman käyttäjämäärää. Tämän toteuttaminen vaatisi kuitenkin Tarmon toimintalogiikan miettimistä laajemmin, jotta se kykenisi ketterästi mukautumaan, sekä perus-, että edistyneempien käyttäjien vaatimuk-

siin. Tämä on kuitenkin otettu alustavasti huomioon nykyisessä arkkitehtuurissa ja tämän idean toteuttaminen ei vaatisi Tarmolta suurta uudelleensuunnittelua.

Laskennan läpinäkyvyyden lisäämisen tulee olla tärkeässä asemassa käyttöliittymän edelleen kehityksessä, koska tällä hetkellä useat kokeneemmat käyttäjät toivovat, että pystyisivät tarkkailemaan laskennan kulkua helpommin. Tämän toteuttaminen vaatii kappaleessa 6.2.1 esitetyn kaltaista semanttista tukimoduulia, joka keskustelee aktiivisemmin laskentapohjien kanssa, antaen käyttäjälle nykyistä enemmän tietoa.

Tämän työn myötä Tarmon dokumentointi on nostettu paljon aikaisempaa paremmalle tasolle. Rakenne on dokumentoitu ja ohjelman koodi on huomattavasti aiempaa paremmin kommentoitu. Sen sijaan laskentapohjien tarkempi dokumentointi (pl. rakenneanalyysi) on yhä kirjavaa. *Laskentapohjien kaavojen, toiminnan ja rakenteen dokumentointi kannattaisi yhdenmukaistaa*, jotta laskentapohjien ylläpito, edelleen kehitys ja laskentojen välisten synergioiden hyödyntäminen olisi helpompaa.

Tarmon *käyttöönotto on syytä tehdä systemaattisella* tavalla. Palautetta tulee kerätä pro-aktiivisesti, ylläpidon pitää vastata käyttäjien ongelmiin nopeasti ja koulutuksen tulee olla hyvin suunniteltu. Nämä kaikki auttavat käyttäjien luottamuksen saavuttamisessa, sekä muutosvastarinnan alentamisessa. Toisaalta on tärkeää myös seurata kuinka alkuperäiset Tarmo-laskelmat ja projektin toteutuneet teholistat korreloivat keskenään.
Laajempi kokemus sekä tämän työn osalta, sekä muiden suunnittelujärjestelmien parista Metsolla on jo aiemmin herättänyt kysymyksen *”äiti”-konfiguraattorin* tarpeellisuudesta. Lukuisat suunnittelua tukevat ohjelmistot ovat tietyllä tapaa konfiguraattoreita, joilla pyritään mallintamaan kyseessä oleva linja luvuiksi, joita laskennat hyödyntävät. Tämä tarkoittaa valtavaa määrä syöttöarvoja, joista merkittävä osa on keskenään päällekkäisiä. Mikäli olisi olemassa ns. äitikonfiguraattori, jolla koko paperikonelinja voitaisiin määritellä ainakin pääominaisuuksiensa osalta, voisi tätä pohjatietoa käyttää kaikissa suunnitteluohjelmistoissa. Tämä vähentäisi manuaalista, kertautuvaa työtä merkittävästi, joka ei itsessään tuota mitään lisäarvoa. Myös syöttöarvojen päivitys revisiosta toiseen helpottuisi, kun tieto pitää päivittää vain yhteen paikkaan, josta se periytyisi myös muihin järjestelmiin. Mielestäni olisi perusteltua tehdä laajempi arvio ”äiti”-konfiguraattorin mahdollisuuksista, sekä haasteista ja pohtia kannattaisiko tällaista konfiguraattoria lähteä toteuttamaan.

LÄHTEET

Abernathy W.J. and Utterback J.M. 1978. “Patterns of Innovation in Industry,” Technology Review, Vol. 80, No. 7, June-July, pp. 40-47.

Basile, F., Chiacchio, P. & Del Grosso, D. 2009. "A two-stage modelling architecture for distributed control of real-time industrial systems: Application of UML and Petri Net", Computer Standards & Interfaces, vol. 31, no. 3, pp. 528-538.

Bluff, R.J. 1999. "Integrated modular avionics: system modelling", Microprocessors and Microsystems, vol. 23, no. 7, pp. 435-448.

Chen, K. & Liu, R. 2005. "Interface strategies in modular product innovation", Technovation, vol. 25, no. 7, pp. 771-782.

Choppy, C. & Reggio, G. 2005. "A UML-based approach for problem frame oriented software development", Information and Software Technology, vol. 47, no. 14, s. 929-954.

Eynard, B., Gallet, T., Nowak, P. & Roucoules, L. 2004. "UML based specifications of PDM product structure and workflow", Computers in Industry, vol. 55, no. 3, pp. 301-316.

Gulliksen, J., Boivie, I. & Göransson, B. 2006. "Usability professionals—current practices and future development", Interacting with Computers, vol. 18, no. 4, s. 568-600.

Halevi, G. 2001. "110 manufacturing methods" in Handbook of Production Management Methods Butterworth-Heinemann, Oxford, pp. 59-310.

Henninger, S. 2000. "A methodology and tools for applying context-specific usability guidelines to interface design", Interacting with Computers, vol. 12, no. 3, s. 225-243.

Immonen, S 2010. “Suunnittelutyökalun käytettävyys ja kehitysprosessi”, Kandidaatin työ, Tampereen Teknillinen yliopisto. 35s.

Isokääntä, I. 2007. “Paperikoneen mekaanisen käytön mitoitus”, Diplomityö, Oulun yliopisto. 118+22s.

Juhola J., Välimaa K. 1997. ”Tuotevarioinnista kilpailukykyä – tarjouksesta toimitukseen”, Metalliteollisuuden keskusliitto, Helsinki, ISBN 951-817-675-2. 80s.

Karjalainen, P. 1998. ”Paperikoneen sähkökäyttöjen mitoitus”, Diplomityö, Oulun yliopisto. 119+70s.

Ketolainen, A. 2002. ”Paperinvalmistusprosessiin tulevien energia- ja vesivirtojen määrittäminen”, Diplomityö, Lappeenrannan teknillinen korkeakoulu. 125+13s.

Ketolainen, A. 2011. Product Vault: ”Material Efficiency”. Metso Paper Oy 2011. Viitattu 30.3.2012. Saatavissa: http://www.metso.com/MP/Marketing/vault2mp.nsf/sets_html/web

Kinnunen, K, 2010. “Metso PBL Line Sales Process”, sisäinen dokumentti, Metso Oy.

Kuo-Min C., Ren-Jye L. 2005. ”Interface strategies in modular product innovation”, Technovation, Volume 25, Issue 7, July, , ISSN 0166-4972, 10.1016/j.technovation.2004.01.013. pp. 771-782

Lehtonen, T. 2011. ”Moduloinnin kurssin luentokalvot”. Tampereen teknillinen yliopisto, 2011.

Marchiori, M. 1998. "Bubbles in modularity", Theoretical Computer Science, vol. 192, no. 1, pp. 31-54.

Mayhew, D. J. 1999. The usability engineering lifecycle: A practitioner's handbook for user interface design. San Francisco, Calif.: Morgan Kaufmann Publishers, Inc., cop.

Metso Annual Report. Metso Oy 2011. Viitattu: 26.3.2012. Saatavissa: http://www.metso.com/reports/2011/assets/pdf/metso_annual_report_2011_english.pdf

Metso Product Vault. 2010. “BelBaie V - OptiFormer shoe blade rebuild sales”-flyer. Metso Paper Oy 2010. Viitattu: 11.8.2011. Saatavissa: http://www.metso.com/MP/Marketing/vault2mp.nsf/sets_html/web

Nielsen, J. 1990. “Paper versus computer implementations as mockup scenarios for heuristic evaluation”. Teoksessa D. Diaper, D. Gilmore, G. Cockton & B. Shackel (toim.) Human-Computer Interaction: INTERACT’90. Proceedings of the IFIP TC 13 Third International Conference on Human-Computer Interaction. Cambridge, UK. 27.–31. elokuuta. Amsterdam: Elsevier, 315–320.

Niskanen, J. 1997. ”Paperiteollisuuden koneet luentomoniste – Kuivatusosa”. Oulun yliopisto, koneensuunnittelun laboratorio. Oulu. 28 s.

Nugroho, A. 2009. "Level of detail in UML models and its impact on model comprehension: A controlled experiment", Information and Software Technology, vol. 51, no. 12, pp. 1670-1685.

OMG. 2005. ”Introduction to OMG’s Unified Modeling Language (UML)”. Object Management Group ltd. Viitattu 5.4.2012. Saatavissa: http://www.omg.org/gettingstarted/what_is_uml.htm

Palomäki, H. & Puurula, M. 2005, ”Investointilaskentasovelluksen käyttäjäkeskeinen suunnitteluprosessi”, Pro gradu-tutkielma, Jyväskylän yliopisto. 141s.

Paulapuro, H. et al. 2000. “Papermaking Part 1 – Stock preparation and wet end”. Jyväskylä, Fapet Oy. 461 s.

Paulapuro, H. et al. 2000. “Papermaking Part 2 – Drying”. Jyväskylä, Fapet Oy. 496 s.

Pirinen, T. 2006. “Press section concepts for paper and board” [verkkodokumentti]. Metso Paper Oy. [Viitattu 01/2007]. Saatavissa: http://intrafiles.metso.com/MP/Marketing/vault2mp.nsf/sets/intra.

Poikonen, M. 2012. Data-ajo metson CRM/referenssi-järjestelmästä, Metso, 2012.

Rathnam, S. & Mannino, M.V. 1995. "Tools for building the human-computer interface of a decision support system", Decision Support Systems, vol. 13, no. 1, pp. 35-59.

Ro, Y., Fixson, S.K. & Liker, J.K. 2008. "Modularity and supplier involvement in product development" in Handbook of New Product Development Management, eds. Christoph H. Loch & Stylianos Kavadias, Butterworth-Heinemann, Oxford, pp. 217-258.

Satzinger, J.W., Jackson, R.B. & Burd, S.D. 2005. “Object-oriented analysis and design with the unified process”, Thomson, Boston. 626s.

Stora Enso. 2011. ”Annual Report”, Stora Enso Oy. Viitattu: 26.3.2012. Saatavissa: http://www.storaenso.com/media-centre/publications/annual-report/Documents/Stora_Enso_Rethink_2011_EN.pdf

Turkki J. 2005, “SC-paperirullien vanaisuuden syntymekanismin selvittäminen”, Diplomityö, Lappeenrannan teknillinen yliopisto. 102+83s.

Ulrich, K. and Tung K. 1991. “Fundamentals of product modularity”, In: Issues in Design/Manufacture Integration, pp. 73-79. A. Sharon Ed. ASME, New York, NY, U.S.A.

Welie, M. 2004. “Patterns in Interaction Design”. Viitattu: 15.6.2012. Saatavissa: http://www.welie.com/patterns/index.php

Willard, B. 2007, "UML for systems engineering", Computer Standards & Interfaces, vol. 29, no. 1, pp. 69-81.