NTNU

Norwegian University of Science and Technology

Master's degree thesis

IP502009 Msc thesis, professional master 90 ECTS

Product Architecture Compressor Starter

1202 / Kai Brudevoll

Number of pages including all pages: 87

Divided into 3 parts

1. Thesis
2. Appendix
3. Research paper

Ålesund, 2.6.2016

Mandatory statement

Each student is responsible for complying with rules and regulations that relate to examinations and to academic work in general. The purpose of the mandatory statement is to make students aware of their responsibility and the consequences of cheating. **Failure to complete the statement does not excuse students from their responsibility.**

*Please complete the mandatory statement by placing a mark **in each box** for statements 1-6 below.*		
1.	**I/we herby declare that my/our paper/assignment is my/our own work, and that I/we have not used other sources or received other help than is mentioned in the paper/assignment.**	☒
2.	**I/we herby declare that this paper** 1. Has not been used in any other exam at another department/university/university college 2. Is not referring to the work of others without acknowledgement 3. Is not referring to my/our previous work without acknowledgement 4. Has acknowledged all sources of literature in the text and in the list of references 5. Is not a copy, duplicate or transcript of other work	Mark each box: 1. ☒ 2. ☒ 3. ☒ 4. ☒ 5. ☒
3.	**I am/we are aware that any breach of the above will be considered as cheating, and may result in annulment of the examination and exclusion from all universities and university colleges in Norway for up to one year, according to the Act relating to Norwegian Universities and University Colleges, section 4-7 and 4-8 and Examination regulations .**	☒
4.	**I am/we are aware that all papers/assignments may be checked for plagiarism by a software assisted plagiarism check**	☒
5.	**I am/we are aware that NTNU will handle all cases of suspected cheating according to prevailing guidelines.**	☒
6.	**I/we are aware of the NTNU's rules and regulation for using sources.**	☒

Publication agreement

ECTS credits: 90

Supervisor: Ola Jon Mork

Agreement on electronic publication of master thesis

Author(s) have copyright to the thesis, including the exclusive right to publish the document (The Copyright Act §2).
All theses fulfilling the requirements will be registered and published in Brage, with the approval of the author(s).
Theses with a confidentiality agreement will not be published.

I/we hereby give NTNU the right to, free of charge, make the thesis available for electronic publication: ☒**yes** ☐**no**

Is there an agreement of confidentiality? ☐**yes** ☒**no**

(A supplementary confidentiality agreement must be filled in and included in this document)

- If yes: **Can the thesis be online published when the period of confidentiality is expired?** ☐**yes** ☒**no**

This master's thesis has been completed and approved as part of a master's degree programme at NTNU Ålesund. The thesis is the student's own independent work according to section 6 of Regulations concerning requirements for master's degrees of December 1st, 2005.

Date: 6.2.2016

Abstract

Developing and designing new product architecture for electrical starters. Questing what system can handle high variety both today and tomorrow’s variety. Methods found in literature are applied. In this thesis technology platform is chosen and developed to prototype stage. Software for handling product structure and documentation developed. Solutions to handle generational variations are suggested.

Acknowledgement

There are many who has helped me during this 3 years at Hials now NTNU. It has been 3 enriching years. I would like to start with thanking Sperre for giving me the opportunity to study part time while I work. Thanks to Ola.J Mork and Irina.E Hansen at NTNU and Stephan Balling at Sperre, they have advised and pushed me to finish this thesis. My father has helped me while writing this thesis, reading the entire thesis to me out loud so we could find alternative words or a better sentence structure. Thank you.

At Sperre the electrical department (Bjørn-Johnny Lorgen & Freddy Alexander Stene) has been a team, having countless discussions, meetings to find new solutions. I'm looking forward to continuing the work after the thesis with you.

Table of content

Innhold

TERMINOLOGY **11**

SYMBOLS **11**

ABBREVIATIONS **11**

BACKGROUND **13**

THEORETICAL BASIS **14**

THE POWER OF PRODUCT PLATFORMS **14**

POWER TOWER MODEL **14**

ULRICH **15**

PRODUCT TOPOLOGY 15

MARTIN DESIGN FOR VARIETY **16**

BRAD WALTON BROOKS THESIS **18**

SUMMERY **19**

METHODS **20**

DEFINING PRODUCT PLATFORM STRATEGY **20**

STEP 1 MARKET SEGMENTS 20

STEP 2 IDENTIFY GROWTH AREAS 20

STEP 3 DEFINE CURRENT PLATFORMS 21

STEP 4 ANALYZE COMPETING PRODUCTS 21

STEP 5 CONSIDER FUTURE PLATFORM INITIATIVES 21

FUNCTION MAPPING **22**

PRODUCTION TIME MEASUREMENT **22**

PRODUCT VARIATION MEASUREMENT **23**

RESULTS **24**

PRODUCT DESIGN **24**

DEFINING PRODUCT PLATFORM STRATEGY 24

FUNCTION MAPPING 29

PHYSICAL LAYOUT 29

MANUFACTURING PROCESSES 30

CONTROLLER 34

VARIETY **37**

SPATIAL VARIATION 37

PLM 43

DISCUSSION **44**

PRODUCT DESIGN **44**

DEFINING PRODUCT PLATFORM STRATEGY 44

FUNCTION MAPPING 47

PHYSICAL LAYOUT 48

CONTROLLER ..... FEIL! BOKMERKE ER IKKE DEFINERT.

SPATIAL VARIETY **49**

PRODUCT STRUCTURE 49

GENERATIONAL VARIETY **50**

CONCLUSION **54**

CONTRIBUTION TO RESEARCH **54**

INDUSTRIAL CASE STUDY **54**

THEORETICAL RESEARCH **54**

METHODOLOGY FOR DESIGN FOR HIGH VARIETY 55

FUTURE WORK **56**

REFERENCE **57**

List of Figures

Figure 1 Power tower model 15
Figure 2 Spatial and generational variety 17
Figure 3 Product function map 22
Figure 4 Documentation steps 31
Figure 5 Production Steps 32
Figure 6 New production steps 33
Figure 7 Touch controller front side 34
Figure 8 Touch controller back side 35
Figure 9 Application running on prototype 36
Figure 10 Documentation Generator 40
Figure 11 Documentation generator flowchart 40
Figure 12 Volume/Tier diagram 44
Figure 13 Function cost diagram 45
Figure 14 Product platform strategies 47
Figure 15 Teamcenter product update steps 52
Figure 16 Example of uploaded dataset 52

List of Tables

Figure 1 Power tower model 15
Figure 2 Spatial and generational variety 17
Figure 3 Product function map 22
Figure 4 Documentation steps 31
Figure 5 Production Steps 32
Figure 6 New production steps 33
Figure 7 Touch controller front side 34
Figure 8 Touch controller back side 35
Figure 9 Application running on prototype 36
Figure 10 Documentation Generator 40
Figure 11 Documentation generator flowchart 40
Figure 12 Volume/Tier diagram 44
Figure 13 Function cost diagram 45
Figure 14 Product platform strategies 47
Figure 15 Teamcenter product update steps 52
Figure 16 Example of uploaded dataset 52

Terminology

Symbols

TPV	Total Product variety [N]
POV	Product range variety [N]
PRV	Product optional variety [N]
||	Logical OR
&&	Logical AND

Abbreviations

CP	Control Panel
BOM	Bill of Material
DOL	Direct On Line
CPMO	Consumer Insights, Product Technologies, Manufacturing Processes
	Organizational Capabilities
OEM	Original Equipment Manufacturer
PLC	Programmable logic controller
PCB	Printed circuit board
Part	(1)Physical component in BOM
	(2) Software, Software parameter, component, I/O and documentation.
CAD	Computer aided design
API	Application program interface
PLM	Product life cycle management
	Old system IBM Notes
	New system Siemens Teamcenter
ERP	Enterprise resource planning system
	Windows Navition system
OSV	
HTTP	Hypertext transfer protocol
HTTPS	Hypertext transfer protocol secure

ITK	Integrated tool kit, Teamcenter API
C#	C-sharp Microsoft programming language

Background

Sperre Industri is a company compressor producer. The company was founded more than 75 years ago. The main costumers are the shipping industry and power plants. Sperre's product range is from the basic bare shaft compressor to complete container with air receivers, dryers, compressors and control-system. These products can be found anywhere in the world.

The company has a strong focus to be a lifecycle partner to the end user. Sperre promise to deliver spare parts to any compressor for 30 years within 2 days.

Sperre has strong product architecture and well defined product ranges for their compressors, but the control system architecture is showing its age. The documentation is stored on an old PLM system. Sperre have not had the capability to program this software for many years. Engineers have added functions, new drawings and new products to the system without following the original rules. This has led to a system that only experience can let you navigate to the correct drawings. The second problem with the current system is that we are using 3 different control technologies and two different platforms. The control systems designs are from the time periods from before 1986, 2000 and 2012. In 2015 the electrical department was asked to develop a new starter for the compressors and move the documentation into a new PLM system. The new PLM system is the Siemens software Teamcenter. Using Teamcenter as documentation storage will let the electrical drawings and the mechanical 3d drawings to use the same system.

Developing the control system for our compressors come with some special challenges. The control system has to interface to our customers control system and the motor that drives the compressor. This leads to a massive amount of variations.

In this thesis I apply methods found in literature to create new product architecture for compressor control systems. The goal of the architecture is to handle the variation met in the market and have a system for the products lifecycle.

Theoretical basis

When starting the thesis I was recommended one book “The power of product platforms “ by Meyer and Lehnerd and a research paper from Ulrich. In addition I did most of my research on google scholar and sci-hub. Google scholar is the easiest source to use, but sci-hub is far superior since it includes all paper and removes any paywalls.

The Power of Product Platforms

The main subject of the book is the power tower model and how to apply different methods to develop product architecture and how to describe a good or bad platform. In this thesis the power tower model and how to define a product strategy are used. Define a product strategy is described under methods. The methods described for measuring how well or bad the platforms perform uses data over several years. We cannot use these methods in my thesis.

Power Tower model

Power tower is a model that describes what goes into a product platform and what market it targets. This is what is called product architecture. See Figure 1 Power tower model.

Market Applications divide the markets into segments and price tiers. Segments can be different customers, geographical areas etc. While Tier represents the price, quality and complexity of the products.

Product platforms are the products you sell, and each platform has many products that share common parts, designs and manufacturing processes.

The common building blocks are

- consumer insights
    - Information that consumers gives to you. This can come from interviews, market research or specifications.
- Product technologies
    - The different technical solutions used in the product platform
- Manufacturing processes

- Steps and techniques used to transform the raw materials into the finished product.

- Organizational capabilities
  - Description of infrastructure in the company. Infrastructure includes distribution network, agents and information systems. Typical information systems are ERP and PLM software.

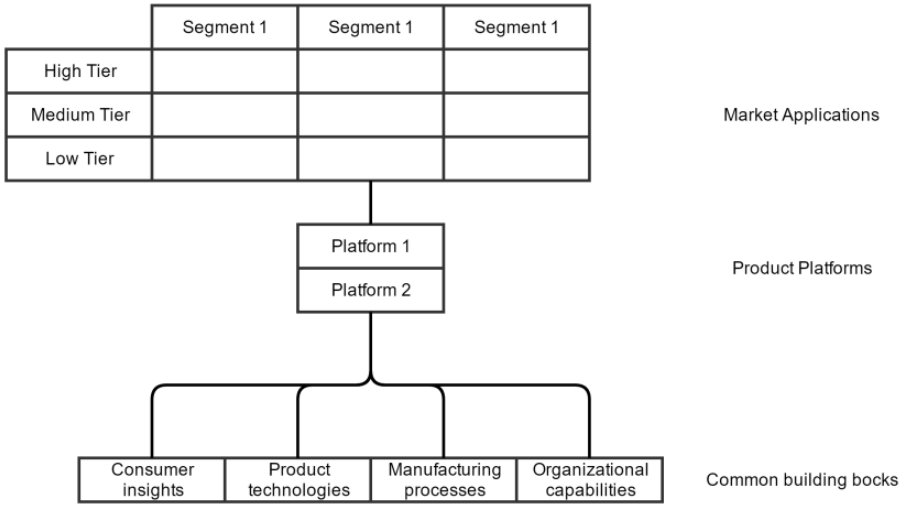


Figure 1 Power tower model

Ulrich

Product topology

Product topology describes how the product is designed. We can divide the product into two main groups, integral design and modular design. Integral designed products cannot change one part without affecting the other parts of the design. While modular designed product has decoupled connections between components of the product. Change of one element does not affect other components. Example of an integral design can be a laptop. If you want to change the screen size you would need to upgrade the entire laptop, while if you had an desktop you can change the monitor without upgrading you entire system.
Modularity comes in 3 types, bus, slot and sectional.

1. Bus

Bus architecture has a common line where new components share the same interface to connect to the rest of the product. Power outlet in your house is a good example. You can connect as many outlets you want on the line.

2. Slot

Each interface is designed for one specific role. Example would be the computer slot for the HDMI cable. There is one interface, but you can connect any type of monitor you want.

3. Sectional

Sectional topology is when the modules used are attached to each other. The modules have a similar interface and a new module can be connected on the previous module. Example of this would be the wooden train set children plays with. One set of track is connected to the next track and so on.

Martin Design for Variety

Martin goes into the topic of product variety in his dissertation. He explains and describes product variation. According to Martin product variety can be divided into two axes: Spatial variation and generational variation. Spatial variation is defined as the product variety offered to the market at a given time, while generational variation is the variation over time. See Figure 2 Spatial and generational variety. This gives us a good description of one of the problems faced when designing new product architecture. We not only need to think of the variation of today, but also the variation of tomorrow.

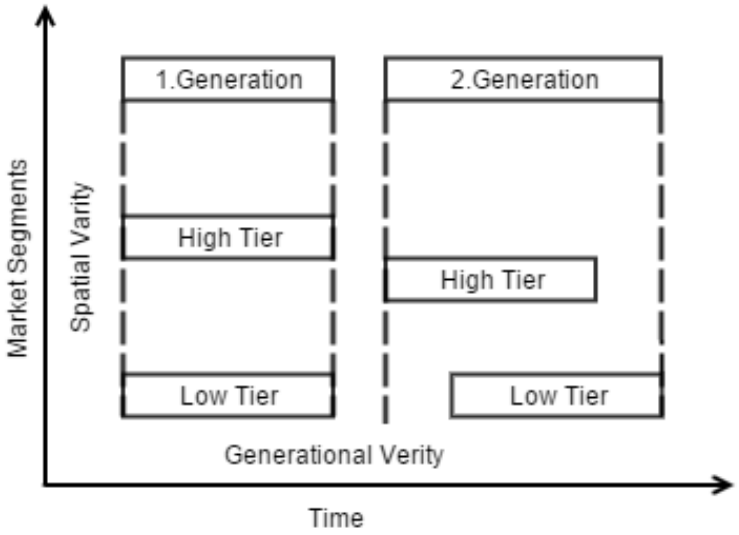


Figure 2 Spatial and generational variety

Generational variation changes over time. The driving factor for change can be many. Martin lists these up. See Table 1 Drivers for generational change.

External drivers
Changing performance needs (weight, ECM, short circuit performance etc..)
Changing environment conditions(temperature, humidity etc)
New functions (new market, new technology)
Reliability improvements
Changing Regulations or standards
Competitor introduction of improved product (Higher quality or lower price)
Obsolescence of parts
Internal drivers
Cost reduction
Reduce assembly time
Reduce component types
Simplifying logistic
Increase meta data
Use lower cost technology
Reduce serviceability requirements
Increase serviceability
Improve component manufacturing process

Table 1 Drivers for generational change

Brad Walton Brooks Thesis

Brad Walton Brooks "Automated data import and revision management in a PLM environment" thesis gives a practical example to implement data import and revision control in Teamcenter. He also gives measurements of possible time savings that can be achieved through automation.

Summery

The papers described above where useful as a theoretical basis or for theirs methods. The theoretical literature gives the vocabulary and an overview of the challenges, while the methods give a path to follow when designing a product architecture. See Table 2 Paper summery.

Name	Author	Source type	Theoretical	Methods
The Power of Product Platforms	March H. Meyer Alvin P.Lehnerd	Book	Yes	Yes
The Role of Product Architecture in the Manufacturing Firm	Karl Ulrich	Research paper	Yes	Yes
Automated Data Import and Revision Management in a Product Lifecycle Management Environment	Brad Walton Brooks	Master Thesis	No	Yes
Design for varity: A methodology for developing product platform architectures	Mark Valeton Martin	Dissertation	Yes	No

Table 2 Paper summery

Methods

Defining product platform strategy

In Power of product platform Meyer and Lehnerd describe a 5 step method to find the platform strategy for your company.

- Step 1 Market Segments
- Step 2 Identify growth areas
- Step 3 Define current platforms
- Step 4 Analyze competing products
- Step 5 Consider future platform initiatives

Step 1 Market Segments

Identify your market, who is buying and what are they buying. Customers are described as marked segments. Segments can be different products, regions, and applications. The goal is to find what differ between your customer groups. Future potential markets should be considered. The product they are buying are divided up in product tiers, where high tier is the most complex product group and low tier is the simplest product group.

Step 2 Identify growth areas

Identify which segments that will grow in the future. There are 5 different key data that needs to be identified.

- The current sales volume
- Your share of each market segment
- Expected 5 year growth rate
- Leading competitor in each segment
- The driving customer-needs in each segment

Step 3 Define current platforms

Define your products into separate platforms. It is not always easy to separate your products into platforms. Some companies have all their products into one platform or several platforms for a single product line. Making high level block diagram can help in the process of finding the commonality of your products. At this step you can find the common systems that can be leveraged over all your products.

Step 4 Analyze competing products

Find the competitors' products and compare them to your own and the other competitor's products.

Step 5 Consider future platform initiatives

We define our strategy and establish our goals for the implementation of the new product platform.

Function mapping

Ulrich describe a method to map functions to physical parts in the product. Physical parts in the product can be different parts used to finish a product. This can be documentation, software or any object that is used to create the product. This gives an overview of the product and give each part one or more function. If a part does not have a function it is not needed. This method can also be used to display the difference between solutions for one function.

See Figure 3 for a single function map with different solutions. I define parts as what we need to change or add to the product to implement the function. The part can be software, software parameter, hardware, I/O or documentation.

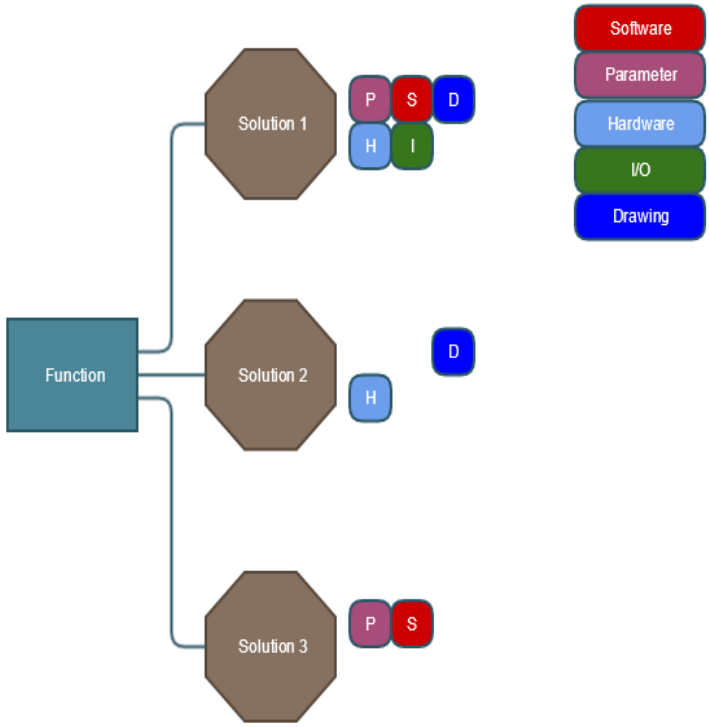


Figure 3 Product function map

Production time measurement

Each step taken during production is described. Start and stop time is noted for each step. The data is shown in a table.

Product variation measurement

Product variation is defined in how many different products you are offering. This gives us a general equation seen in Equation 1. V_T is the total variation while V_n is the sum of mutually exclusive options. If option 1 and option 2 is mutually exclusive then we gather them into the variable V_1 and give V_1 a value of 3.

Equation 1 Variation

$$V_T = V_1 * V_2 * V_n$$

Results

Product design

Defining product platform strategy

Step 1 Market Segments

What we want to know is who are the customer and what do they buy. Who are the costumers are defined as market segments. Market segments are easily defined at Sperre since we already have them defined. These are E&P, Marine, Power plant and Nitrogen. Then we need to find what the customer buys. Product are described as low tier, medium tier or high tier, but Sperre does not have a clear definition for tiers
We choose to use the number of input and output signals from various instruments as the defining factor for placing each product into a tier. These instruments can be drain valve as output or temperature switch as input.

We define a low tier starter as 1 input signal and 1 output signal. X represent input while Y represents output.

Equation 2 Low Tier

If ((1=<X=<1)||(1=<Y=<1)) then Low Tier

We define medium tier starter as more or equal to 2 to 3 input signals, and more or equal to 2 to 3 output signals.

Equation 3 Medium Tier

If ((2=<X=<3)||(2=<Y=<3)) then Medium Tier

We define high tier as more than 3 input or more than 3 output signals.

Equation 4 High Tier

If ((3<X)||(3<Y)) then High Tier

Name	Signals		Tier
	Input	Output	
Switchboard type 1	1	1	Low
Switchboard type 2	3	1	Med
Switchboard type 3	3	1	Med
Switchboard type 4	3	2	Med
Switchboard type 5	3	1	Med
Switchboard type 6	1	0	Low
Switchboard type 7	2	1	Med
Starter X	5	1-2	High

Table 3 Tier definition

The result are shown in Table 3

Retrieving data

First we need to retrieve the data from Sperre ERP system Microsoft Navision. This is not straightforward since we need to compare two lists to get the data we require. We use the "Finished Prod. Orders" and "Prod card" page in Navision and apply filters seen in appendix

Then we run the Marketsegment program to merge the two tables. When two line has the same order number we merge the two datasources. This is processed and exported to a result text file. Marketsegment program can be seen in Appendix 4.

Step 2 Identify growth areas

Sales volumes are kept internal for Sperre, but normalized data is showed in Table 4Table 4 Results. Table 4 shows that the marine is the dominant marked and medium tier is dominant range.

Growth rate is a harder number to quantify. Short term marine segment is retracting in numbers. Larger ships like tankers, container and small fishing vessels are being ordered and built while dry bulk and support vessels are not being built.

Inn 2015 our market share of the global marine market was 20% (source Sperre)

	Relative in segment			Relative to total		
Segment	High Tier	Medium Tier	Low tier	High Tier	Medium Tier	Low tier
EP	70%	30%	0%	1.6 %	0.7 %	0.0 %
Power Plant	48%	48%	3%	3.3 %	3.3 %	0.2 %
Marine	21%	61%	18%	18.6 %	54.5 %	15.9 %
Internal	6%	69%	25%	0.1 %	1.3 %	0.5 %
			Sum:	23.6 %	59.8 %	16.6 %

Table 4 Results

The marine segment is 89% of our total volume. The marine market has several different class authorities. Class authorities have different requirements to the control system and starter. See Table 5 Class requirements.

Class	Temperature	Oil level	Tier level
LR	X	x	Medium
GL	X	x	Medium
BV	X		Medium
RINA	X	x	Medium
NK		x	Medium
KR		x	Medium
DNV			Low
CSS			Low
CR			Low
ABS			Low
RMRS	X	x	Medium

Table 5 Class requirements (2)

In addition I found information on how much each class represents. See Table 6 Class Volume. Result is normalized to each class and total.

	Normalized to class			Normalized to total			
Class:	High Tier	Medium Tier	Low Tier	High Tier	Medium Tier	Low Tier	Sum
ABS	16%	67%	18%	4%	16%	4%	25%
DNV GL	27%	47%	26%	7%	12%	7%	26%
BV	17%	76%	7%	2%	10%	1%	14%
LRS	26%	56%	18%	3%	5%	2%	10%
CCS	19%	62%	19%	1%	4%	1%	6%
Other	28%	49%	7%	6%	10%	1%	17%

Table 6 Class Volume

Step 3 Define current platforms

Starter product platforms in Sperre is defined as classic and X starters. Classic platform has been in use since 1986 and has been upgraded twice. Once to Logo PLC relay replacement and once to an embedded solution with a membrane interface. Classic product platform has a very high variety and legacy problem. Classic starters are used for the classic compressor platform.

X starters are the newest product platform. The X range starter is a controller used for the X compressor platform and is based on an embedded solution with a membrane interface technology. In total we have 3 different control technologies. See Table 7 Platform/Competitior table.

Step 4 Analyze competing products

Competitors

Obtaining data from the marine market is fairly easy. The marine market is a transparent market that is focused by many papers and economical institutions. Our leading competitors in the marine segment are the German company Sauer compressors. For the other segments we don’t have a clear picture. However in our last competition we competed against Boge compressor.

Technologies

Relay control is the simplest form of control. Relays are wired in a way to create logical control. Relay control is used to solve the simplest form of control problems.

PLC has a wide range from the small relay replacement to large industrial solutions. Generally PLCs are used “for one off” solutions where flexibility is valued above price.

Embedded is a custom made PCB with a microcontroller or processor. Embedded is more commonly used by OEMs.

Membrane interface is a plastic screen with integrated buttons and led`s. The screen usually has printed graphics and symbols.

Industrial touch screens are flat screens with a transparent resistor matrix.

See appendix 1 for comparison between our platforms and leading competitors.

Name/competitor	Tier	Technology
Switchboard type 1	Low	Relay control
Switchboard type 2	Med	PLC control
Switchboard type 3	Med	PLC control
Switchboard type 4	Med	PLC control
Switchboard type 5	Med	Relay control
Switchboard type 6	Low	Relay control
Switchboard type 7	Med	Relay control
Starter X	High	Embedded+ Membrane Interface
Sauer	Low	Relay control or plc
Sauer	High	Embedded+ Membrane Interface
Boge	High	Embedded + Touch & M’ Interface

Table 7 Platform/Competitior table

Step 5 Consider future platform initiatives, is found under discussion.

Function Mapping

Ulrich describe how to map each function to specific parts in your product, while Meyer & Lehnerd describe how to map out the complexity of your product. I have tried to combine both by indexing a fixed score cost to each part used. I use the loose definition of parts as Ulrich use in his description. In my index parts are defined as:

- Software — New software is needed
- Software parameters — Parameter can be changed in standard software
- Hardware — Physical component has to be added to BOM
- I/O — Input or output is used on the controller
- Documentation — Documentation is changed

Each part gets a base cost, where software parameters are cheaper to implement than software and new hardware has the highest cost. Each function is showed in rows and given a number value in each part. The function Alarm requires minimum two relays with relay one holding relay and one lamp this is showed by giving the hardware cell a value of 2 and documentation a value of 1. If the cell is blank the function does not require a change in the specific part. This is done for all 4 different technologies. Relay control, PLC embedded with membrane interface and embedded with touch screen.

The function map can be seen in Appendix 2

Physical Layout

In appendix 9 to 10 you can see the layout of the starter. This layout will be standard for all starters supplied by Sperre. Inside the enclosure we have divided the space into 3 sections. First section is the high power section down to the left. Terminals are located down to the left. The terminals has standard numerical markings and are divided into digital inputs, analog inputs and voltage free relay signals. Costumer will guide his signal cables into the cable conduit to the left

and connect to the terminals. Each I/O signal from the controller can be configured on the touch screen. Last sections have the smaller parts used like transformer, emergency run switch and add-on parts.

The front of the starter has a standard layout. Only ampere meter and main switch can be changed between different ratings and types.

Manufacturing processes

Manufacturing process in Sperre can be divided into two parts, documentation and production.

Documentation old way

When the sales department gets a PO they convert their "Quote" into an order. This order appears in the "!QA order list" in the ERP system. The electrical engineer starts the process seen in Figure 4. Engineer check the order for attributes like voltage, frequency, required power for compressor, power available from motor, RPM, motor nominal current and required functions. Functions, voltage, frequency and nominal motor current are used to find the correct starter. If this starter has not been made before, the engineer makes the documentation and stores it for future use. Time used to create new documentation varies. This can take one- to 8 hours depending on the complexity of the starter. Documentation steps vary depending on the customer and the requirement for each project. Some orders jumps straight to spes-drawing. These drawings are often a small change compared to our normal standard. One example would be a customer using a different color system for their signaling system. Most system uses green as running signal "all good it is running". Then there is the alternative red as running signal "be aware machine is running". These cases are often easiest handled manually for each case.

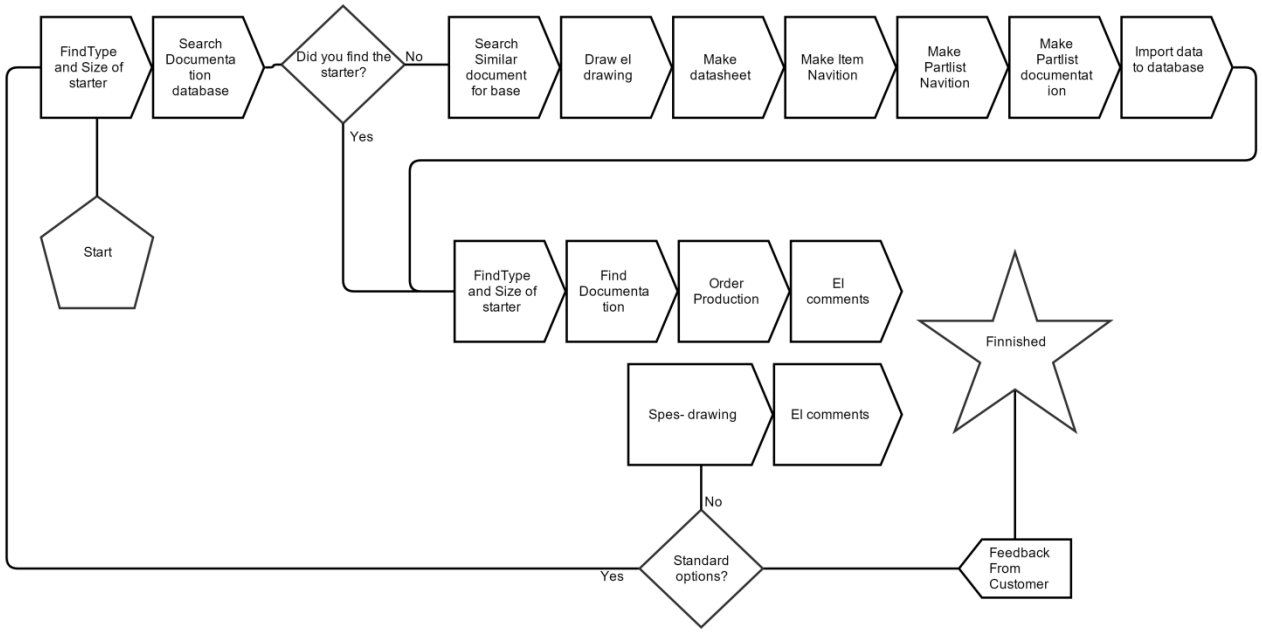
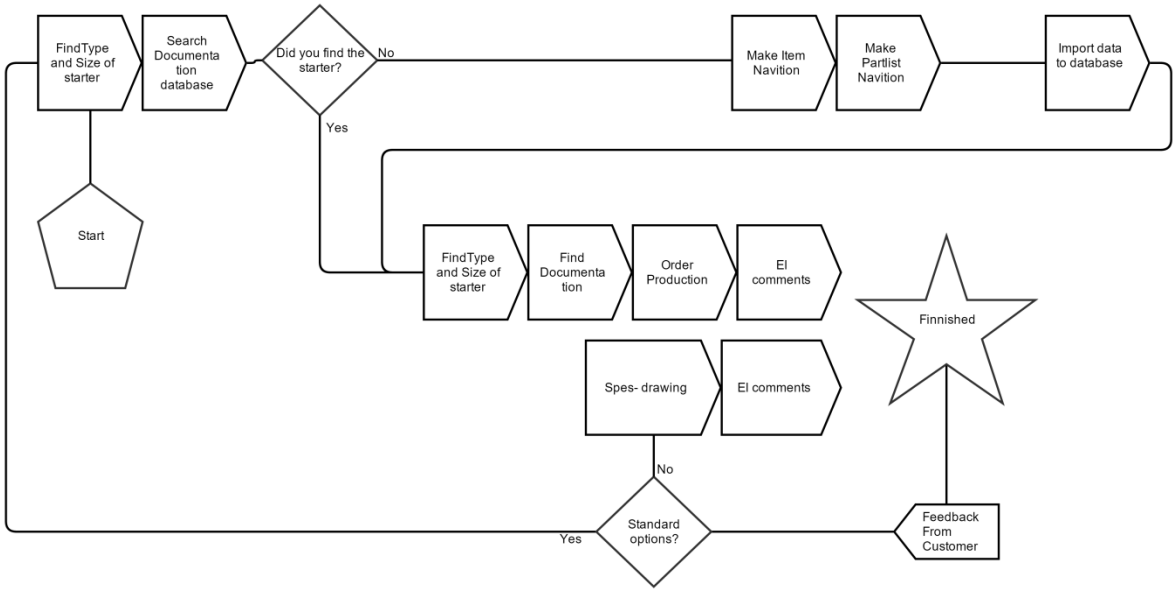


Figure 4 Documentation steps

Documentation new way

We can simplify the process steps by automating the generation of the electrical drawing, datasheets and parts list. The steps that are needed to create new documentations are removed and replaced with a function selector and a generate documentation button. ERP system is still manual and needs to be created for each new configuration used.



Production old way

Production steps are well-known steps in the company and can be seen in Figure 5. The Forman has a priority list of production orders to be fulfilled. The list is updated each week by the production manager.

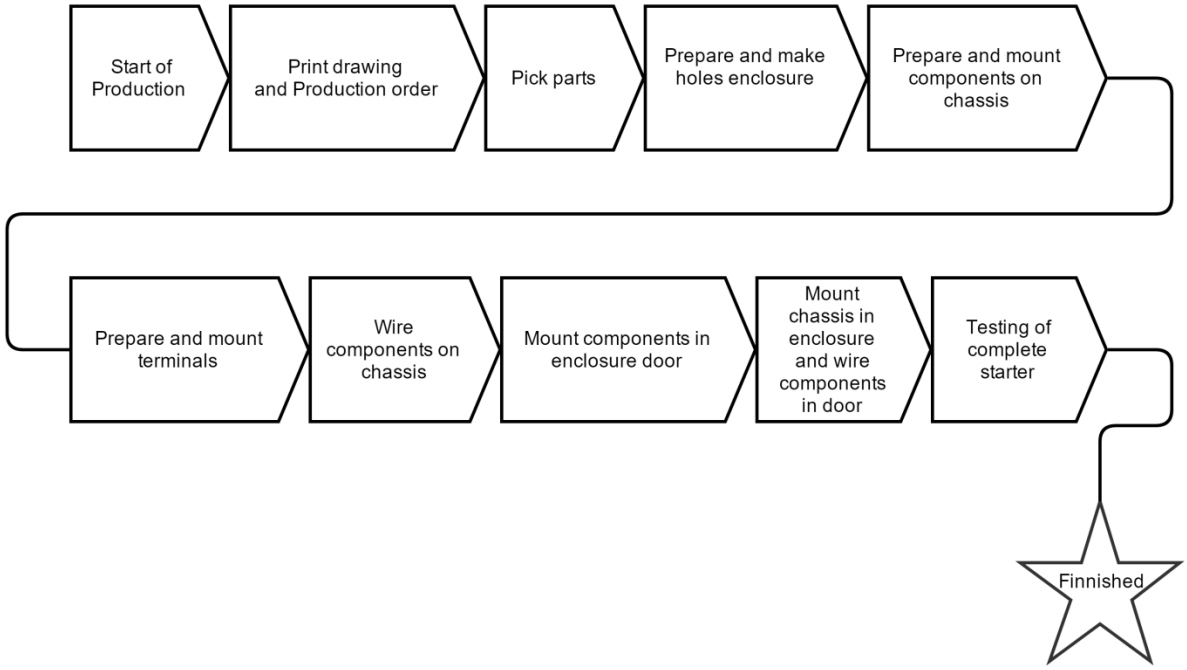


Figure 5 Production Steps

The time measurement was measured by the Forman. Each operation was not done by the same person since operators has their preferred tasks. All operators can handle each step except the first and last step. Printing drawings and production order require knowhow to use the ERP system. The last step is the quality control of the starter. All functions and components are tested. If a fault is found this is repaired. The result from production time can be seen in Table 8.

Step	Time Used	Time Used	
Start of Production			
Print drawing and production order	0:15:00	0:15:00	Operator 1
Pick Parts	0:16:00	0:29:00	Operator 1
Prepare and make holes enclosure	0:33:00	0:33:00	Operator 2
Prepare and mount components on chassis	0:19:00	0:19:00	Operator 2
Prepare and mount terminals	0:14:00	0:14:00	Operator 2
Wire components on chassis	1:40:00	1:10:00	Operator 3
Mount components in enclosure door	0:10:00	0:10:00	Operator 3
Mount chassis in enclosure and wire components in door	0:45:00	0:45:00	Operator 3
Testing of complete starter	1:20:00	1:20:00	Operator 4
Finished production total time	5:32:00	5:15:00	

Table 8 Production time measurement

Production new way

By using a touchscreen as a HMI we can standardize the cutout of the enclosure. When ordering enclosures you can order the cutout for a small extra price. By taking advantage of this offer we can take away one step of the process. Reducing assembly time by 33 minutes and reducing total assembly time by 11%. New process can be seen in Figure 6.

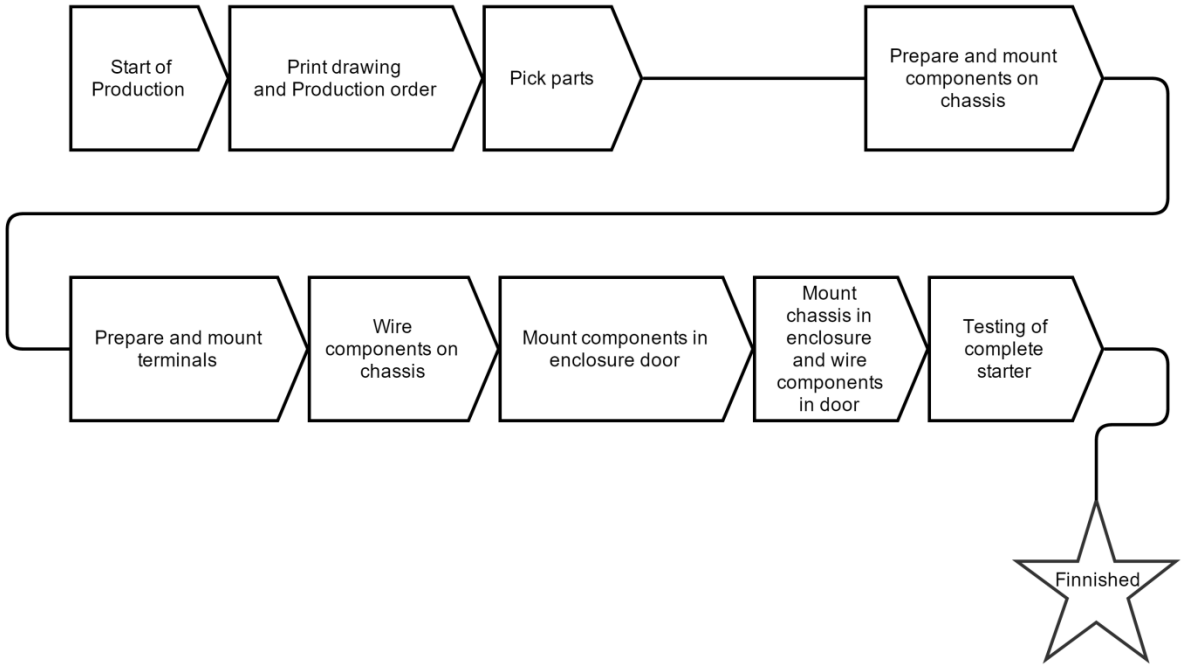


Figure 6 New production steps

Controller

In December 2015 Sperre signed a contract to develop a new controller with Data Response. This was the results from comparing different solutions from different suppliers. See appendix 1. Data Respons had the best combination of cost and technical solution.

Work method

Data Respons designs the electrical schema and writes the software. The hardware placement is designed in Taiwan and the production is located in Taiwan. Software and hardware specifications are developed in cooperation with Sperre, Taiwan and Data Respons.

The development is not finished during the writing of this thesis, but the first prototype has been delivered.

Hardware

The brain of the controller is a Cortex ARM CPU. The interface is a 7" touch screen. For storage there is a 8Gb memory card. For I/O there are 6 Analog inputs, 8 Digital inputs and 8 relay outputs. These I/O's are placed on the backside of the controller.
Additional ports are found on the side of the controller. These are 1 USB connection, 1 mini USB, 1 RS485 two wire and a RJ45 10/100/1000Mb connection.

Figure 7 Touch controller front side

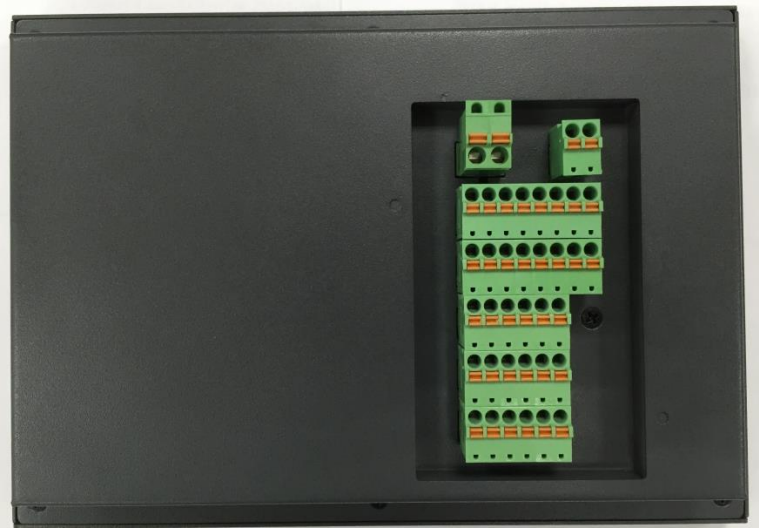


Figure 8 Touch controller back side

Software

The controller runs on Yocto based Linux distribution. The distribution includes QT c++ library and the drivers to run the I/O.

The application shown to the customer is running on top of the operating system. The application is written in c++. The QT GUI is a modern interface which is often found in different embedded applications.

The application will control the compressor. The machine state diagram is shown in appendix 12. There are two main modes of operation, auto mode and manual mode. In auto mode the compressor will keep the pressure in the air receiver between two pressure points. In manual mode the air receiver will be filled once, then stop. This will be standard for all compressors.

Figure 9 Application running on prototype

User levels

There are 3 levels of users. First level is used to control parameters that control the running of the compressor. Second level is called service level and is used by the chief or other technical personnel. Service level can change parameters that affect the interface between the controller and other systems. The highest level is Sperre level. This level has full access to the controller. The Sperre user should have full knowledge of how to configure the controller.

Parameter

The cheapest way to handle variety is handled by parameters. The application has several parameters that can change pressure settings, I/O configurations and optional control. The function “Motor heater” is a good example. If the heater output is activated the interface gets a new blue lamp, and one relay is used to control the heater element.

Alarms and shutdown values can be changed by service level user, while the Sperre level user can change the range the service level user can modify.

Language pack is included in the software, but only English is implemented at the start.

Variety

Spatial variation

PLM and part number old way

In our old PLM system starters are organized in 3 levels. First level is for the type of starter. The next level is divided between functions and options. The second level we store software and electrical drawings. In the last layer we store the part list.

- Starter type 2
    - Functions (Drawing & software)
        - Part list 60Hz + Ampere and Voltage Attributes
        - Part list 50Hz

Part list is divided into two main groups 60 and 50 HZ. This is represented in the part number of the starter. There is no system other than incremental use of numbers. If a new voltage/current range is made the next free number is used. See Table 9.

	Starter Type	Function	Frequency	Voltage/Current Range
No of Char	2	3	1	2
Example	E2	202	1	12

Table 9 Old part number system

In appendix 11 I compare the customer options and the total combination this leads to. Our old starters have a total possible variation of approximately 15 million. This number is in truth higher since we have 7 different starter series.

The Equation 1 Variation is used.

PLM and part number new way

All product documents are stored in Teamcenter PLM program. Teamcenter can stores files and metadata. Teamcenter has 4 main datatypes: Folder, item, revision and dataset.

Folders work as normal Linux folders that can crosslink and have recursive calls. In our system they are organized into a normal Windows folder tree. Item is one starter. The Item is identified with the metadata “ItemName”. “ItemName” has the same value as the part number in our ERP system. Under each Item we find at least one revision of the starter. For each change in the product a new revision is created. The documentation files are called datasets in Teamcenter. These are found under each Revision.

- Folder
    - Item
        - Revision
            - Dataset

Folders structures are organized into 2 levels. Voltage range and then starting method.

- Voltage Range
    - Starting method
        - Items

Part number system has 6 levels and 8 characters. First character is used to sort the part number in our ERP system. This is done so we don’t mix starters with other items. The rest of the system can be seen in Table 10. First level we describe voltage range and starting method. This is

described with 1 character. Second level describes current range. Level 3 to 6 describes up to 20 functions. By using the triacontakaidecimal system, also known as extended hex system, up to 5 functions can be described for each character used. Each function is either active or not active. Functions with 2 or more exclusive options uses more bits. If function 1 is activated and 2-5 is deactivated, this will be represented as 1. If function 1 and 2 is active, this is represented as 1+2=3.

	Primary Atributes		Options																			
	Volt & Start Method	Ampere	Function 1-5					Function 6-10					Function 10-15					Function 15-20				
Partnumber	0-9	00-99	0-V					0-V					0-V					0-V				
Bit			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
			Function 1	Function 2	Function 3	Function 4	Function 5	Function 6	Function 7	Function 8	Function 9	Function 10	Function 11	Function 12	Function 13	Function 14	Function 15	Function 16	Function 17	Function 18	Function 19	Function 20

Table 10 New part number system

Documentation generator

See attachment and press one of the two links to see video of the generator working. The documentation generator program has an interface where each attribute range and function is listed. When selecting your desired attributes and function the part number is displayed at the bottom. For sales persons only this aspect of the program is available, but for the electrical engineer there is additional option to generate documentation. When pressing the generate button the program generates all the required documentation files.

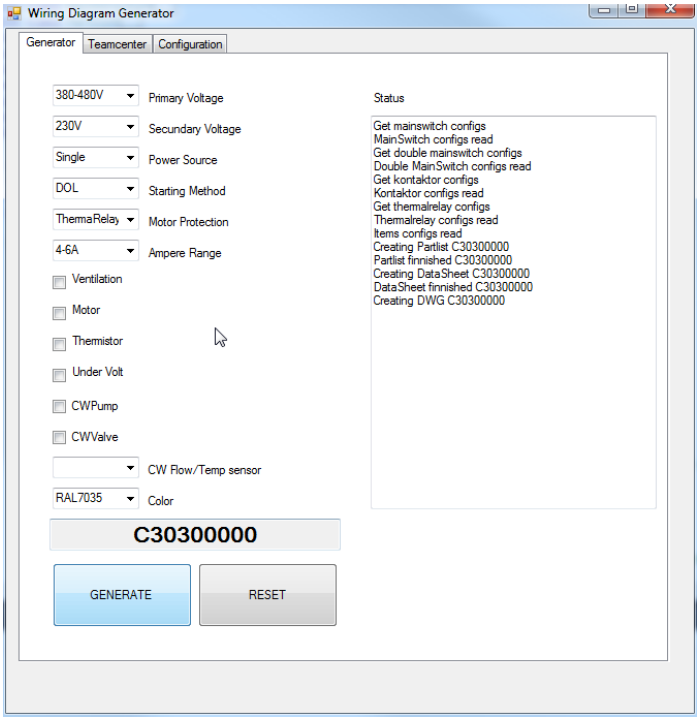


Figure 10 Documentation Generator

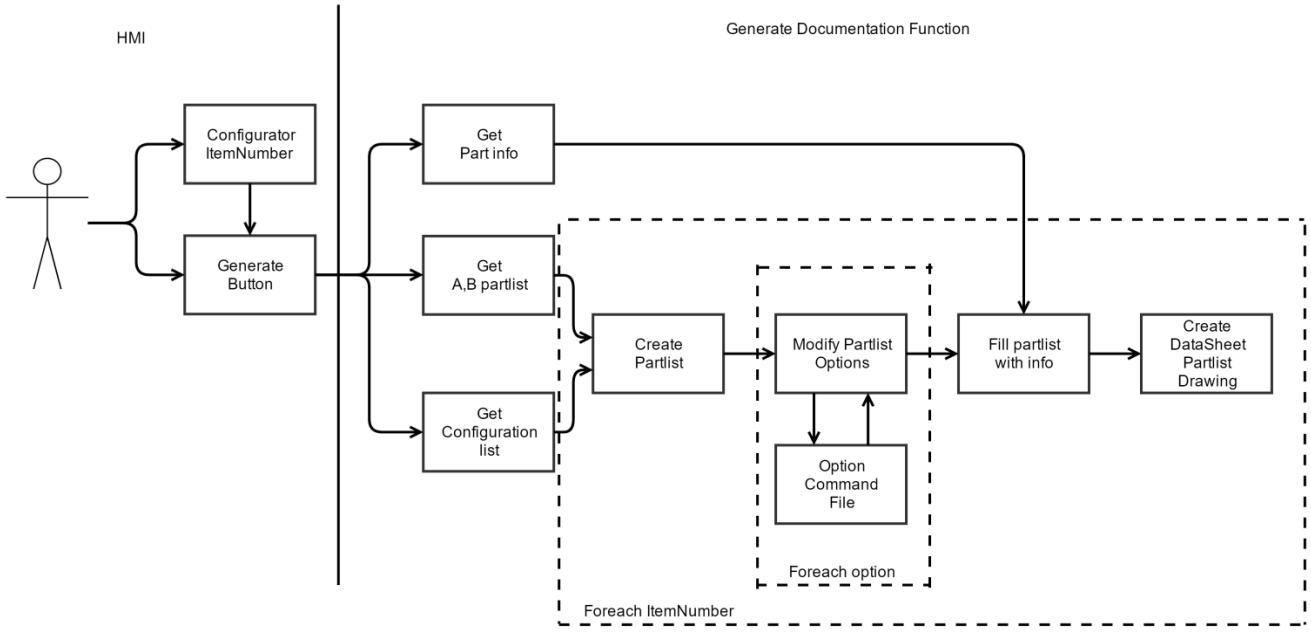


Figure 11 Documentation generator flowchart

Wiring diagrams

To create electrical diagrams we start by making one electrical diagram. This diagram is the one diagram all other diagrams get created from. We draw each option in separate layers. For example we have 3 layers for motor voltage [230, 380-480, 600-690]. Only one layer can be active at a time. In addition layers also have a "not activate layer". Ref. appendix 13. Where relays are not used the layers needs to be replaced with a short-circuit.

Bricscad is a cad program. It has a .net api drivers for windows. When using the api you can start Bricscad in the background and open the electrical diagram. The program sends instruction through the API to modify which layers are visible and which layers are not visible. At last it modifies version number and diagram name before exporting the diagram to PDF. DWG are copied to output folder.

Part lists and datasheet

When making the product structure we first make class for different parts used. Different parts have different attributes that affects the final attributive of the product.
The size of the main-switch, contactor and thermal relay decides the size of the motor the starter can control. Attributes can also affect the attributive of the part. Voltage, ambient temperature affects the rating of components. This information can be stored in the different items classes. The item class is a general class used for parts that does not affect or is affected by the main attributes of the product. Each part is checked when making a part-list. The worst attribute is used to describe maximum rating for the complete starter.

Items classes:

- Contactor
- Thermal Relay
- Softstarters
- Mainswitches
- Item

The next step is to find the common parts used in the product range. Dividing them into 3 categories. In category A we have put the most common parts like terminals, controller, cabinet size etc… Category B the parts that affect main attributes like ampere, voltage and starting method. Contactors, thermal relays, softstarter, transformers and main switches are in B category. Category C is used for different options. Category C has two functions: Add and Replace. Add function simply appends a part to the part-list. Replace function search the existing part-list and replace the part. An option like 115V control voltage needs to change all 230V parts to 115V. This means that each option for C has a list of all items that gets replaced and with what item it gets replaced with.

There are 4-6 different A lists and 45 different B lists. These lists are combined with a combination list where B BOM is linked to a A BOM. When the customer wants to add options we replace or add items found in C list. In appendix 12 both A, B and a combination matrix can be seen.

Bill Of Material				
A1		B2		C
Part 1	+	Part 21	+/-	Replace Part 21 with Part 31
Part 2		Part 22		

Table 11 Bill of material

After the part list has been created. The parts in the list are filled with data from Item classes. Attributes like weight, ingression protection, ampere rating and power rating are found. One datasheet with all attributes are exported to pdf. Two part lists are exported one to tab delimited text for ERP import, the other are exported to PDF format. Parts with the legend equal “Blank” are not shown in this PDF. End user only needs to know the major parts of the system and the parts most likely to need replacement during its life time.

Files

When all the files have been generated we have the following files. See attachment for examples for each of these files.

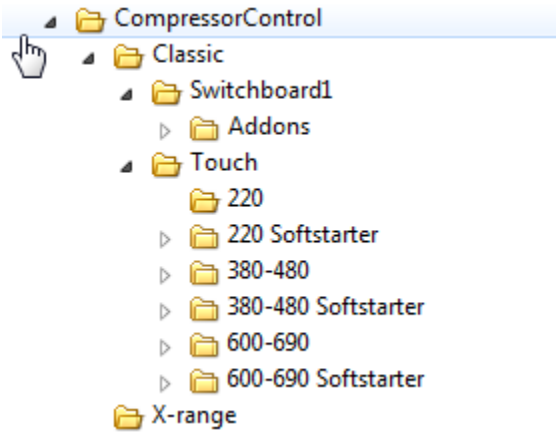
- Wiring diagram PDF
- Datasheet PDF
- Part list PDF
- Wiring diagram DWG
- Part list TXT

ERP

In the ERP system each part number is defined as an item. The item contains price, cost, weight, production department, replenishment method, product groups and BOM. Each starter item has an BOM where all parts used for the production of one starter is included. This BOM is used to create an “Pick” during production.

PLM

We store the documentation in Teamcenter. From teamcenter the entire organization can access the documentation. The starters are organized in a folder structure.



What system to handle revisions is not chosen. I have converted the methods used by Brad Walton Brooks to .net except the release item function. The code is based from Teamcenter .net examples and expanded to include upload data. Examples where I run a query, create item, create revision and upload data. The test has been conducted to our PLM system.

Temacenters api files TCSOA dlls are used as an library.

Discussion

Product design

Defining product platform strategy

Customer insight

From our market segment research we find some interesting results. We find that Sperre sells mostly ABS and DNV classed starters. DNV requires low tier starters, but only 26% of the starters are placed in the low tier category. This shows us that the market wants more complex starter than the class authorities requires.

60% off all the starters produced are medium tier. This would argue for an optimization off our product for medium tier, and then take low and high tier into consideration.

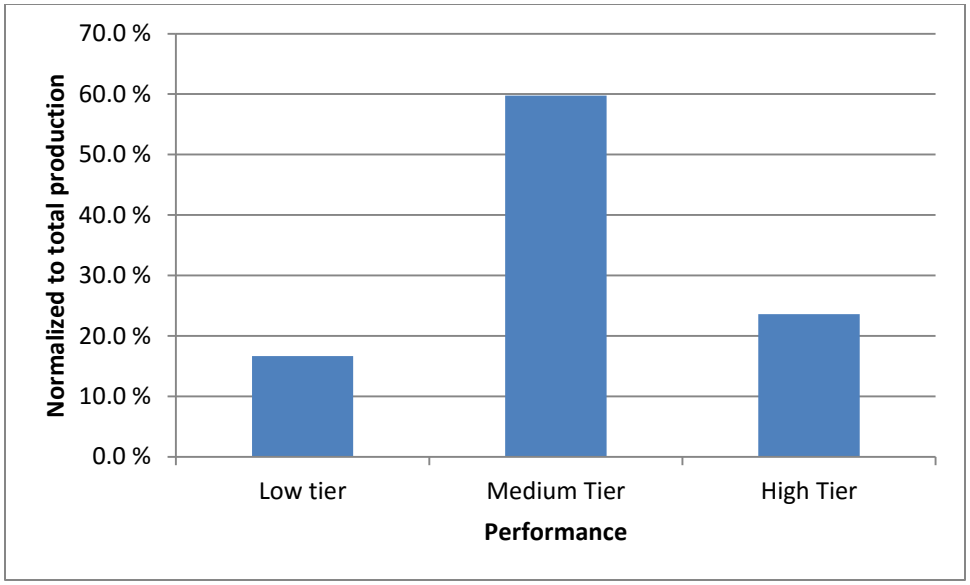


Figure 12 Volume/Tier diagram

Technology

In Figure 13 Function cost diagram we see the cost when comparing different technologies. The score represent a cost. The calculation can be seen in appendix 2. The score is derived from how each function is implemented. The minimum number of functions for a starter is 7. We realize

that relay control has the smallest initial cost, but has a steep curve and end up being one of the most expensive one. Low tier usual end up around 7 or 8 functions, while Medium tier starts at 10 functions. From 9 functions and up, the touch technology has an advantage over relay control. Touch interface has a slight advantage over membrane interface.

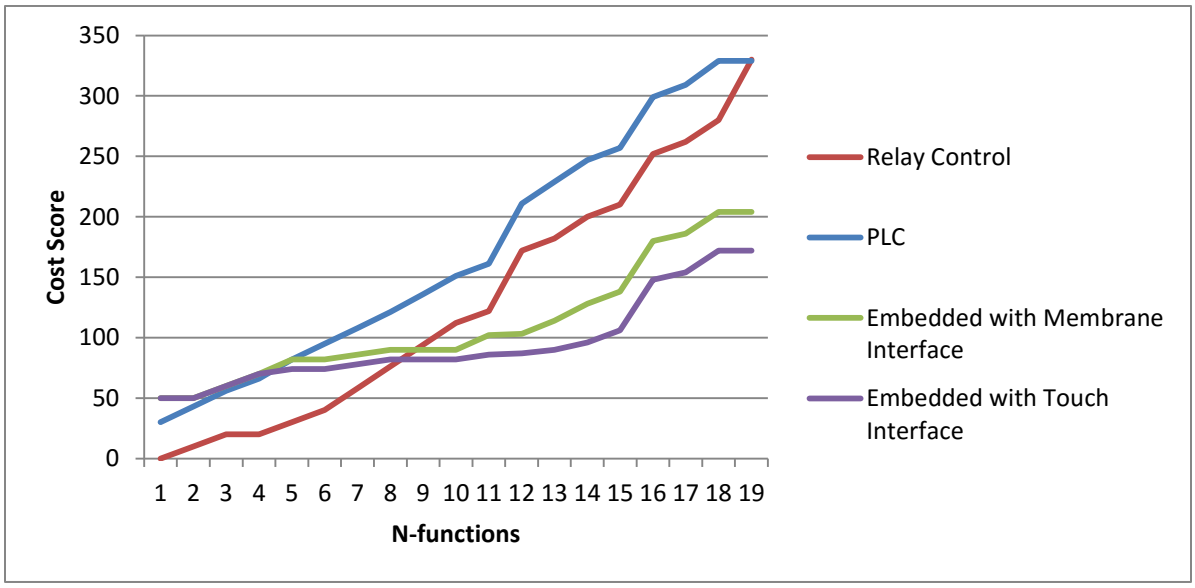


Figure 13 Function cost diagram

Manufactory process

The manufacturing process can be divided into two parts, documentation phase and production phase. Both phases are essential to the final product. These two phases are part of a bigger flow of Sperre, but this thesis is only about the starter. The steps for each of the phases are described under results.

Documentation

There are two standards to optimize the documentation work, strict standards or include all solutions as standard. Strict standards would not work for Sperre since the products often are integrated into other systems and standards. Most customers send their specifications and require their sub vendor to find a solution, which complies with their specifications. The second way is to include all solutions as a standard. We have implemented all solutions as a standard for our X-starter platform. This has led to less CAD work, but more support mails to customers explaining their different options and/or if an option is included in their order or not. For the motor a heater

is an extra option, while for the starter the circuit to control a heater is standard. After delivery one typical question is: Where are the connections to the motor heater? The customer assumes the heater is included in the order, since the electrical wiring diagram show the heater, but the heater is an option and not a standard delivery. Installing a heater into a motor after the motor has been delivered from the sub-vendor is expensive. My solution is to generate wiring diagram and documentation from software. This combines the good sides of customizations and including all options. The customer gets documentation that only includes what he has ordered. One of the bad sides of generating documentation is that it requires a certain familiarity and knowledge of programming. A partial solution is to use excel and tab delimited txt files as an interface between the electrical engineer and the need for programming.

Production

By using touch screen instead of switches or membrane interface we can simplify the production. Adding a new interface like a switch or lamp can be programmed into the touch screen instead of physical adding switches and lamps to the product. When adding a switch or a lamp the enclosure needs additional holes. This makes it impossible to standardize the cutout of the enclosure. With touch screen we can standardize the cutout and order an enclosure that is pre-cut to our specification. Quotes to sub vendors have shown that the extra cost for enclosures with cutout is less than our production cost. Table 8 shows that by removing cutout we can save 33 minutes from our production time.

Outcome

Meyer and Lehnerd recommends to maximize horizontal leverage and vertical scaling when creating a platform strategy. The best textbook strategy is to design one starter for low tier then vertically and horizontally scale the product. This is not optimum for us. As shown in Figure 13 the most optimum solution for low tier starters scales badly vertically.

The other solution is to optimize the product for medium tier and scale down, but this leads to higher cost for the low tier product. Low tier products are very cost sensitive compared to higher tiers and we will risk losing market share.

By splitting our product platform into two distinct platforms we can optimize the cost for low, medium and high tier starters. The low cost solution can use the simplest off control technique's "relay control", while medium and high tier use embedded controller with touch screen. In Figure 14 we can see platform 1 is the touch screen architecture and platform 2 is the relay control solution.

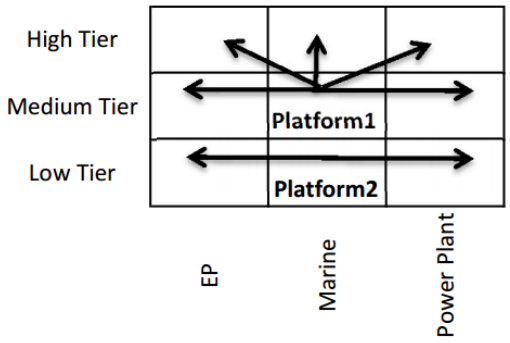


Figure 14 Product platform strategies

Function mapping

Function mapping shown in appendix 2 is a cross combination of Ulrich function mapping with Meyers complexity cost. The descriptions of how the method is applied see results.

I found this data was best represented by using excel table. Both Meyer and Ulrich prefer using a map to represent the cost or complexity. Tables give numbers that can easily be represented in a graph. The weakness of using a table is that N-Functions have a given vector along the X axis. Each function is a sum of all the previous functions and you don't really know the sequence of functions the costumer wants. In my case I can use my experience to sort the functions in a sequence that somewhat will represent reality.

The second bias is that alarms and shutdown function is only represented as one alarm or one shutdown. Reality is that some starters have several alarms and shutdowns. The graph is in reality steeper for relay control than it's represented in Figure 13. Since relay control are most optimal for low amount of N functions and touch interface for more complex solutions, this does not impact my conclusion.

Physical Layout

In appendix 16 and 17 you can see the layout of the starter. Compared to the layout of the old starters we have done some innovations.

- Moved the main switch to the right, from center.
- Rotate terminal list 90 degrees and moved it from down right to the left side.
- Standardized the terminal list
- Standardized the cutout

By moving the main switch to the right of the contactor we can utilize more of the space. The starter is divided up in power zones. Earlier we had motor voltage to the left of the main switch and control voltage to the right. If needed we cannot move the main switch, since it has a door handle. If we move the main switch it will no longer fit the cutout of the enclosure. Often there is more space available on the control voltage side than the power side. We can now utilize this space by shifting the components to the left if needed.

Turning the terminal list 90 degrees gives the electrician more space to terminate his wiring. We also gain more space to fit longer strips of terminals.

For terminals we choose to include all options and give it an abstract description. Options are handled in documentation and the hardware is standardized. By having a standard terminal list we can outsource the production of the terminal list and open the possibility to use standardized wiring harnesses.

We get two advantages by standardizing the cutout. Saving 33 minutes production time and a cleaner workshop, by avoiding metal leftovers.

Spatial variety

The old product architecture was divided into 7 different starter platforms. Which starter that was chosen depended on the sensors or valves used on the compressor. Each starter platform has a standard starter and the customer can add different custom solutions to this starter. In appendix 11 we can see the variety of the options are up to 15 million different configurations. There are more options, but I have removed options that have not been used the last 4 years. In the appendix you also can see the new configuration and there are 3240 different configurations.

Product structure

Old

In the old system all starters where organized in 3 levels. First level is for the type of starter, the next level is divided between functions and options. The second level we store software and electrical drawings. The last layer we have the BOM. BOM is divided into two main groups, 60 and 50 HZ.

- Starter type 2
    - Functions (Drawing & software)
        - Part list 60Hz (Ampere and Voltage Attributes)
        - Part list 50Hz

The old system has several weaknesses.

1. Software can often be the same for different drawings, but since they are stored in the same place as the drawing, there are several copies of the same software in different locations. Updating software becomes a difficult task.
2. Sorting functions/ customer options before attributes as ampere and voltages creates extra work. Changing ampere or voltages changes many components and the physical size of the product, while adding a heater switch does not affect the rest of the product.
3. The last stage where the part lists are stored are not organized. Only incremental numbers are used. This means 10-15 ampere range with 440V can be followed by 50-80 ampere range 690 volt. There are configurations that can have more than one valid part list for a given motor. This

is the result when an engineer looking for the correct part-list and overlooks the correct part-list. He then creates a new part-list and a duplication of the correct part-list, he was supposed to use.

4. Historically part-lists used to include motor and sensors. Part-list level is cluttered by old documents which are no longer valid.

New

We start by finding the primary attributes that decide which starter we choose to use. Those attributes are motor voltage, current and starting method.

The first axis of variation is the primary attribute "voltage". The voltages used worldwide are 230,380,400,415,440,460,480,500,600,660 and 690. From these we find 3 ranges. 230, 380-480 and 600-690 volt. This gives us the first 3 variants. We choose to handle 500V as a special case since most components don't have 500V rating and the market is limited.

Starting method is the second primary attribute we chose to handle. We have two types of starting methods; Direct On Line (DOL) start and soft-start. This gives us 2 x 3 = 6 variants. The last primary attribute current comes in different ranges depending if soft-starting or DOL method is used. DOL has 10 different ranges and soft-starter has 5 different ranges. This gives us a total variant 3 x 10 + 5 x 3 = 45. With these 3 primary attributes we have our base product. Now we can start adding options available to the customer that affect documentation or the BOM. These functions are heater, thermistor, remote start, munsell color, custom color and 110V control voltage. The customer can select several options at the same time. This gives us a combination logic where time 2x2x2x2x3x45 = 2160 possible spatial variants. Note that color is mutually exclusive where RAL7030, munsell and custom color are the available options.

Generational variety

The second type of variety is the generational variety. Over time new revisions of the product are generated. There are several reasons for this listed in Table 1. How can we handle changes and keep track of old documentation at the same time keeping engineering time to a minimum.

In the old system each starter and each BOM had an individual revision number and was manually upgraded when the starter type was sold. This works for a time, but problem arises when you have several updates. Let's say you have 10 revisions since the base line product was launched. I call this a global revision since it affects all starters. You need to use an old starter that has its individual revision 2. You don't know what global revision the old starter has been updated to. You have to check if each global update has been applied to the old starter. If you don't check you risk sending outdated drawings to production.

One alternative is to keep a global revision system and a local revision system. The global revision system should keep track of all changes. If we change the circuit of the motor heater, this would affect also starters without a heating circuit. If there is an error in a single drawing the local revision is updated. Using the same example as above, we are at revision 10 and the old starter we need to use has global revision 2. We can check the global revision list and apply the latest 8 revisions. In Table 12 this solution is referred to as option 1.

Brad Walton Brooks discuss and implements an automated data import and revision control for our PLM system Teamcenter. He comes with a solution and tests the solution. If we can combine the automated documentation generation and data import tools to Teamcenter, we can implement a new revision and push updates to all starter documentations. See Figure 15 Teamcenter product update steps.

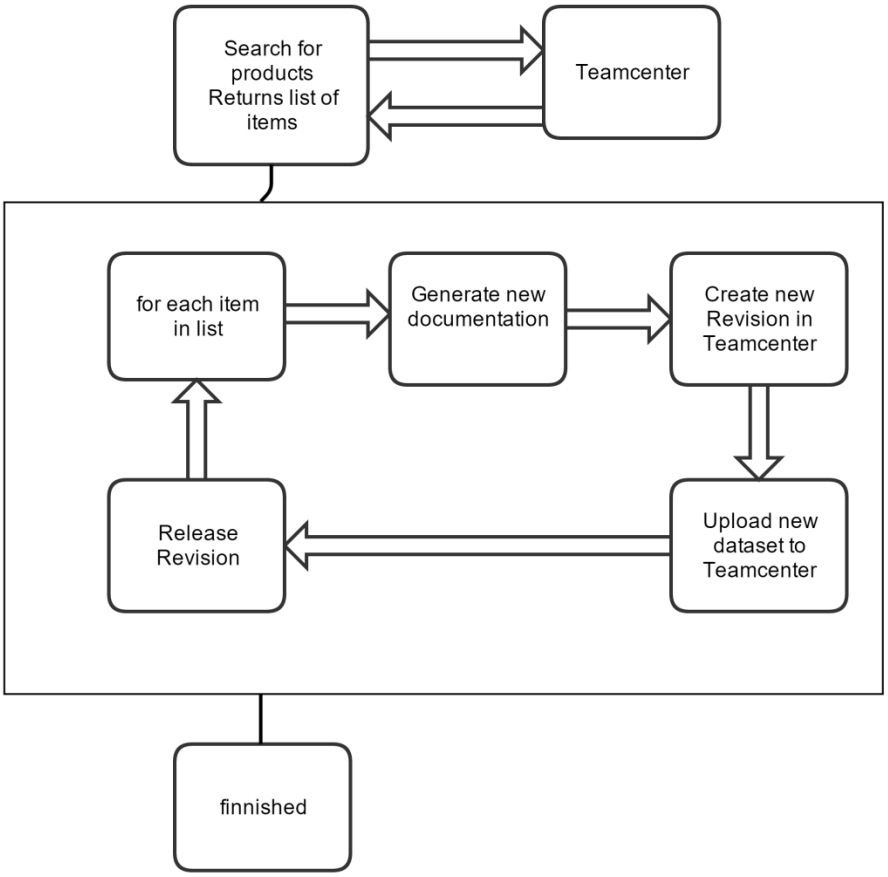


Figure 15 Teamcenter product update steps

Walton describes different functions to communicate with Teamcenter, that we need to convert to C#. The documentation generator program is written in C#. Teamcenter has an API for C#. The same functions Walton describes works for C#. The API communicates over HTTP calls instead of IKT. All functions except release revision functions has been converted and tested to work. This comes under future work that will be continued after the thesis delivery.

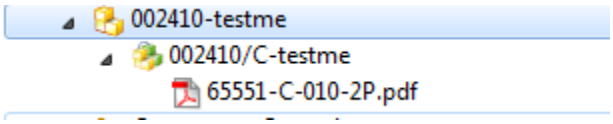


Figure 16 Example of uploaded dataset

Walton has measurements of time savings. He has 3 runs of updating item revision, upload data and release revision. The operation takes an average time of 23 minutes for each item. Today we

have over 800 different drawings for starter1-7, if we had to update all the drawings this would take 306 hours to update. In Table 12 this solution is referred to as option 2.

Solution 1 Manual individual Update		Solution 2 Automated Teamcenter Update	
PRO	CON	PRO	CON
Easy to implement	Constant update work	Easy to maintain up-to-date documentation	Hard to implement
	Old documentation are easily used		Not easy to maintain the system
	2 types of revisions.		Large amount of data created.

Table 12 Revision control system

Conclusion

The new product architecture will give Sperre a flexible platform to develop new functions and features for the future. The software developed to create the documentation will save many engineering hours and increase the flexibility to do changes across the range. The work described in this thesis is the start, but in a few years all electrical control systems can be incorporated into the same software and product architecture. This will keep the department free and flexible to solve future problems instead of struggling with the old legacy system.

Contribution to research

Industrial case study

Both Meyer and Ulrich are product architecture authors and their methods are valid. The use of the method "product architecture strategy" from Meyer was very useful to organize the thought process and creating goals for the new products. I would recommend others to use his clear step by step guide.

Ulrich I found more obvious. His contribution is probably more taught and known, and therefore I found most of his concept fairly well known. Even so the paper "The Role of product architecture in the manufacturing firm" gives a good introduction and a view into the terminology used and methods to consider.

Theoretical research

Mark Valeton Martins gives a good description of variation and generational variation. This gives a good window into the problems and challenges that you need to consider. I did not find his DVT method to be applicable to my case. I did not find any papers on how to deal with large amount of variety. Probably I have been using the wrong search queries, since I have been concentrating more on results than research. I made a short description of the steps I used to create the software to generate the product variation. See "Methodology for design for high variety".

Methodology for design for high variety

1. Item number
    a. Find the main attributes. Main attributes are the attributes that affects the product the most. Normally this would be attributes that scales the product to a bigger or smaller size.
    b. Sort the main attributes and describe them with one or more character use this for the Item number
    c. Create a list of all options the customer is offered. Sort them into groups that cannot be combined. Use this list to make a binary map of each option.
    d. Convert binary map to a different number system. Octal, Hex, hex32 or hex64 are good alternatives. Append these characters to the end of the item number.

	Options																			
	Function 1-5					Function 6-10					Function 10-15					Function 15-20				
Hex extended	V					V					V					V				
Decimal	16	8	4	2	1	16	8	4	2	1	16	8	4	2	1	16	8	4	2	1
Binary	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Options	Function 1	Function 2	Function 3	Function 4	Function 5	Function 6	Function 7	Function 8	Function 9	Function 10	Function 11	Function 12	Function 13	Function 14	Function 15	Function 16	Function 17	Function 18	Function 19	Function 20

Decimal	Hex32	Decimal	Hex32	Decimal	Hex32
0	0	11	B	22	M
1	1	12	C	23	N
2	2	13	D	24	O
3	3	14	E	25	P
4	4	15	F	26	Q
5	5	16	G	27	R
6	6	17	H	28	S
7	7	18	I	29	T
8	8	19	J	30	U
9	9	20	K	31	V
10	A	21	L		

2. Create a product configurator application where you can select each option and show the item number.
3. Parts
    a. Start by defining all parts and their attributes in a list. Attributes like part number, description, weight, dimensions and price.
    b. List of BOM. The BOM contains only part number. The BOM can be multilevel. If more than 1 BOM is combined then a combination table is needed.
    c. For each option create a list. Each line contains a replace or add command. Replace command can be used to replace or delete a component. Replace has two values "search for" and "replace with". Call a method for each option that read the option list and modifies the BOM according to the option list.
    d. Fill BOM with attributes from step 3a.
4. Generate Documentation
    a. Create part lists and datasheets from the BOM. Sum up all attributes like weight, price to get total value. Use worst or best attribute found for other attributes.
    b. If needed create documentation from CAD. Most CAD software has an API that can communicate with your software.
5. PLM
    a. This part goes under future work.

Future Work

When the thesis is finished the first prototype of the controller has been received. There is a lot of work remaining to finish the first product line. After the first product line is finished the work to incorporate the rest of the starters into the system.

For research I would feel there are lacking theories or methods under product architecture for designs that holds an infinity numbers of variations and how to apply this to both generational and spatial variation.

Reference

[1] The Role of product architecture in the manufacturing firm – Karl Ulrich
Massachusetts Institute of technology, Sloan school of Management. Dec.1993
Research paper

[2] The power of product platforms - Marc H. Meyer & Alvin P. Lehnerd
The free press 1997.
Book

[3] Design for varity – Mark Valeton Martin
Department of mechanical engineering and committee on graduate studies of Stanford University in partial fulfillment of the requirements for the degree of doctor of philosophy Nov.1999
Dissertation

[4] Automated Data Import and Revision Management in a Product Lifecy –Brad Walton Brooks
Brigham Young University – Provo. Sept.2009
Master Thesis

[5] Sperre technical guide

[6] Services guide
Siemens SOA API documentation.

Appendix

1. Competitors and platforms comparison 2
2. Function map 3
3. Market segment table filters 4
4. Market segment program 5-10
5. Options and spatial variant calculations 11
6. Bill of material type A and B example 12
7. Bill of material type C example 13
8. Combination table 14
9. Documentation generator 15
10. Starter Layout front 16
11. Starter layout inside 17
12. Machine state diagram 18
13. Documentation printout wiring diagram 19
14. Documentation printout part list 20
15. Documentation printout datasheet 21

Video link
Off documentation generator

Video link (Youtube)
Off documentation generator

Competitors and platforms comparison

Platforms		
Manufactorer	Sperre	
Platform	X-Range	
Technology	Embedded + Membrane Interface	
Manufactorer	Sperre	
Platform	Classic-Range	
Technology	PLC	
Manufactorer	Sperre	
Platform	Classic-Range	
Technology	Relay	
Manufactorer	Sauer	
Platform	unknown	
Technology	Embedded + Membrane Interface	
Manufactorer	Boge	
Platform	Focus Control	
Technology	Embedded + Touch Screen + Membrane Interface	

Function map

Function cost compared to different technology solution

	Color Pink	
Software	Red	2
Software Parameter	Pink	1
Hardware	Yellow	8
I/O	Green	3
Documentation	Blue	2

	Relay Control					Score	PLC					Score	Embedded						Touch					
Functions																								
Initial cost						0						30						50						50
Start/Stop			1		1	10			1	1	1	13						0						0
Emergency stop			1		1	10			1	1	1	13			1		1	10			1		1	10
Emergency start						0			1		1	10			1		1	10			1		1	10
Auto/Manuel			1		1	10			1	2	1	16		1	1	1		12		1		1		4
Remote/Local			1		1	10			1		1	10		1	1	1		12		1		1		4
Reset			1		1	10			1	1	1	13						0						0
Alarm			2		1	18			1	1	1	13		1		1		4		1		1		4
Shutdown			2		1	18			1	1	1	13		1		1		4		1		1		4
Seperate Alarm			2		1	18	1		1	1	1	15						0						0
Seperate Shutdown			2		1	18	1		1	1	1	15						0						0
Setpoint selection	Not Possible					50	Not Possible					50		1				1		1				1
Lead/Follow mode			1		1	10	1		1	2	1	18		1	1		1	11		1			1	3
Normal/Eco mode			2		1	18	1		1	2	1	18			1	2		14				2		6
Normal/Emergency power			1		1	10			1		1	10			1		1	10			1		1	10
Auto change over			5		1	42			5		1	42			5		1	42			5		1	42
Motor heater			1		1	10			1		1	10		1		1	1	6		1		1	1	6
Motor thermistor			2		1	18	1		2		1	20			2		1	18			2		1	18
Service interval	Not Possible					50						0						0						0
						330						329						204						172

Market segment table filters

Field	Filter
Due Date	01.01.14..31.12.14
Source Type	Item
Prod department	ELEKTRO
Description	STYRE*|START*|SWIT*|2W*|1W*

Table 1 Finished Prod. Orders Field Filters

Field	Filter
Due Date	01.01.14..31.12.14

Table 2 Prod card Field Filters

Market segment program


```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace MarketSegments
{
    class Program
    {
        static void Main(string[] args)
        {
            var p = new Program();
            int i = 0;
            string line;
            String Ordernr, Startertype;
            List<String[]> File1 = new List<String[]>();
            List<String[]> File2 = new List<String[]>();
            List<String[]> File3 = new List<String[]>();
            string filepath = "C:\\Marketsegments Analytics\
              \FinnishedProdorder.txt";
            string filepath2 = "C:\\power tower\\Marketsegments Analytics\
              \Prodcard.txt";
            string fileout3 = "C:\\Marketsegments Analytics\\result.txt";

            System.IO.StreamReader ConfigFile = new System.IO.StreamReader
              (filepath);
            while ((line = ConfigFile.ReadLine()) != null)
            {
                if(i>5)
                {
                    File1.Add(line.Split('\t'));
                }

                i++;
            }
            i = 0;
            ConfigFile.Close();
            ConfigFile = new System.IO.StreamReader(filepath2);
            while ((line = ConfigFile.ReadLine()) != null)
            {
                if (i > 3)
                {
                    File2.Add(line.Split('\t'));
                }

                i++;
            }
            ConfigFile.Close();
            bool State = false;
            i = 0;
            foreach (String[] File1SS in File1)
            {
                Ordernr = File1SS[2];
                Startertype = File1SS[3];
```

```
            foreach (String[] File2SS in File2)
            {
                if(File2SS[1].Equals(Ordernr))
                {
                    i++;
                    string Tier = p.GetTierLevel(File1SS[1]);
                    File3.Add(new string[] { Ordernr, Startertype, File1SS[4],
                     File1SS[1], File1SS[5], File2SS[2], File2SS[10], File2SS[5],
                     File2SS[9], File2SS[11], Tier });
                    State = true;
                    break;

                }


            }
            if (!State)
            {
                File3.Add(new string[] { Ordernr, Startertype, File1SS[4],
                 File1SS[1], File1SS[5], "", "", "", "", "","" });
            }
            State = false;

        }

        //result list
        string Writeline = "";
        System.IO.StreamWriter Writefile = new System.IO.StreamWriter
          (fileout3);
        foreach (String[] File1SS in File3)
        {
            foreach(String Linje in File1SS)
            {
                Writeline= Writeline+ "\t"+Linje;
            }
            Writefile.WriteLine(Writeline);
            Writeline = "";
        }
        //Segments
        int[] EP = new int[5] { 0, 0, 0, 0, 0 };  //EP, EPTOT, HIGH, MED, LOW
        int[] Powerplant = new int[5] { 0, 0, 0, 0, 0 };  //Powerplant, EPTOT,
          HIGH, MED, LOW
        int[] Marine = new int[5] { 0, 0, 0, 0, 0 };  //Marine, EPTOT, HIGH,
          MED, LOW
        int[] intern = new int[5] { 0, 0, 0, 0, 0 };  //intern, EPTOT, HIGH,
          MED, LOW


        foreach (String[] File1SS in File3)
        {

            if(File1SS[5].Equals("INTERNAL"))
            {
                intern[0]++;
                intern[1] = intern[1] + Int32.Parse(File1SS[4]);
```

```
                    if(File1SS[10].Equals("High"))
                    {
                        intern[2]++;
                    }
                    if (File1SS[10].Equals("Medium"))
                    {
                        intern[3]++;
                    }
                    if (File1SS[10].Equals("Low"))
                    {
                        intern[4]++;
                    }
                }
                if (File1SS[5].Equals("POWERPLANT"))
                {
                    Powerplant[0]++;
                    Powerplant[1] = Powerplant[1] + Int32.Parse(File1SS[4]);
                    if (File1SS[10].Equals("High"))
                    {
                        Powerplant[2]++;
                    }
                    if (File1SS[10].Equals("Medium"))
                    {
                        Powerplant[3]++;
                    }
                    if (File1SS[10].Equals("Low"))
                    {
                        Powerplant[4]++;
                    }
                }
                if (File1SS[5].Equals("MARINE"))
                {
                    Marine[0]++;
                    Marine[1] = Marine[1] + Int32.Parse(File1SS[4]);
                    if (File1SS[10].Equals("High"))
                    {
                        Marine[2]++;
                    }
                    if (File1SS[10].Equals("Medium"))
                    {
                        Marine[3]++;
                    }
                    if (File1SS[10].Equals("Low"))
                    {
                        Marine[4]++;
                    }

                }
                if (File1SS[5].Equals("EP"))
                {
                    EP[0]++;
                    EP[1] = EP[1] + Int32.Parse(File1SS[4]);
                    if (File1SS[10].Equals("High"))
                    {
                        EP[2]++;
                    }
```

```
                if (File1SS[10].Equals("Medium"))
                {
                    EP[3]++;
                }
                if (File1SS[10].Equals("Low"))
                {
                    EP[4]++;
                }
            }

        }
        Writefile.WriteLine("\n\n\n=====//EP, EPTOT, HIGH, MED, LOW");
        foreach (int l in EP)
        {
            Writefile.Write(l + "\t");
        }
        Writefile.WriteLine("\n=====//Powerplant, EPTOT, HIGH, MED, LOW\n");
        foreach (int l in Powerplant)
        {
            Writefile.Write(l + "\t");
        }
        Writefile.WriteLine("\n=====//Marine, EPTOT, HIGH, MED, LOW\n");
        foreach (int l in Marine)
        {
            Writefile.Write(l + "\t");
        }
        Writefile.WriteLine("\n=====//intern, EPTOT, HIGH, MED, LOW\n");
        foreach (int l in intern)
        {
            Writefile.Write(l + "\t");
        }
        Writefile.WriteLine("\n\n\n CLASS");


        // Class

        List<Program.ClassUsed>  classlist = new List<Program.ClassUsed>();
        Program.ClassUsed Class;
        Class = new ClassUsed();
        Class.name = "DNV";
        Class.numberof = 0;
        Class.numbLow = 0;
        Class.numbMed = 0;
        Class.numbHigh = 0;
        classlist.Add(Class);


        foreach (String[] File1SS in File3)
        {
            State = false;
            foreach (Program.ClassUsed C in classlist)
            {
                if(File1SS[9].Equals(C.name))
                {
                    C.numberof++;
                    State = true;
```

```
                    if(File1SS[10].Equals("High"))
                    {
                        C.numbHigh++;
                    }
                    if (File1SS[10].Equals("Medium"))
                    {
                        C.numbMed++;
                    }
                    if (File1SS[10].Equals("Low"))
                    {
                        C.numbLow++;
                    }

                    break;
                }

            }
            if (!State)
            {
                Class = new ClassUsed();
                Class.name = File1SS[9];
                Class.numberof = 1;
                if (File1SS[10].Equals("High"))
                {
                    Class.numbLow = 0;
                    Class.numbMed = 0;
                    Class.numbHigh = 1;
                }
                if (File1SS[10].Equals("Medium"))
                {
                    Class.numbLow = 0;
                    Class.numbMed = 1;
                    Class.numbHigh = 0;
                }
                if (File1SS[10].Equals("Low"))
                {
                    Class.numbLow = 1;
                    Class.numbMed = 0;
                    Class.numbHigh = 0;
                }
                classlist.Add(Class);
            }
        }
        foreach (Program.ClassUsed C in classlist)
        {
            Writefile.WriteLine(C.name + "\t" + C.numberof + "\tHighTier:\t"
              +C.numbHigh + "\tMediumTier:\t" + C.numbMed + "\tLowTier:\t" +
              C.numbLow);
        }



            Console.ReadKey();
    }
    public string GetTierLevel(String Startertype)
    {
```

```
            string Tier = "";

            if (Startertype.Contains("Switchboard type 1") || Startertype.Contains
              ("Switchboard type 6") || Startertype.Contains("Switchboard  type
              6"))
            {
                Tier = "Low";
            }
            if (Startertype.Contains("Switchboard type 2") || Startertype.Contains
              ("Switchboard type 3") || Startertype.Contains("Switchboard type 4")
              || Startertype.Contains("Switchboard type 5") || Startertype.Contains
              ("Switchboard type 7"))
            {
                Tier = "Medium";
            }
            if (Startertype.Contains("Starter"))
            {
                Tier = "High";
            }

            return (Tier);
        }
        public class ClassUsed
        {
            public string name;
            public int numberof;
            public int numbHigh;
            public int numbMed;
            public int numbLow;
        }

    }
}
```

Options and spatial variant calculations

Costumer options	Old Variants	New Variant	Comments
220V main voltage with thermal relay	10	10	No change
220V main voltage with softstarter	10	5	Softstarter with motor protection used
220V main voltage with MCB	10	0	Shot circuit protection option removed
380-480V main voltage with thermal relay	10	10	No change
380-480V main voltage with softstarter	10	5	
380-480V main voltage with MCB	10	0	Shot circuit protection option removed
600-690V main voltage with thermal relay	10	10	No change
600-690V main voltage with softstarter	10	5	Softstarter with motor protection used
600-690V main voltage with MCB	10	0	Shot circuit protection option removed
Sum Ampere range:	**90**	**45**	Sum of the Variation of the atributes Starting metdod, protection and voltage
Main Isolation Switch	**3**	**3**	No change
Main Isolation Switch for two power supply sources			
Door operated main switch	**2**	**1**	Door handle standard
Power on signal led lamp	**2**	**1**	Light in screen has the same function as led lamp
Common alarm signal led lamp	**2**	**1**	Standard alarm lamp in HMI
remote emergency stop	**2**	**1**	Standard with jumper as terminal bypass
local lead /follow switch for two compressors	**3**	**1**	I/O configurable
local lead follow switch for three compressors			
function for remote lead/follow for two compressors	**3**	**1**	I/O configurable
function for remote lead/follow for three compressors			
local / remote switch for remote control function	**2**	**2**	No change
heating element inside panel	**2**	**1**	Removed as an option and will be handled on case by case order
anti condensationheating element control	**2**	**1**	I/O configurable
compressor oil level control	**2**	**1**	I/O configurable
Junction box	**2**	**2**	No change
Ral 7030 color	**3**	**3**	No change
Munsell color			
Color spesific color			
110V control voltage	**2**	**2**	No change
60 Hz frequency	**2**	**1**	System can handle both 50 and 60 hz
	14929920	3240	

Bill of material type A and B example

BOM type	3B6	
Nr	Legend	ItemNr
1	K1	66225
2	F1	66314
4	P1	8050-02
5	T1	805-050
6	T2	67114
7	Q1	66110
8	blank	66170

BOM type	A1	
Nr	Legend	ItemNr
1	blank	1260002
2	blank	24431-4
3	blank	24431-5
4	blank	2490197

Bill of material type C example

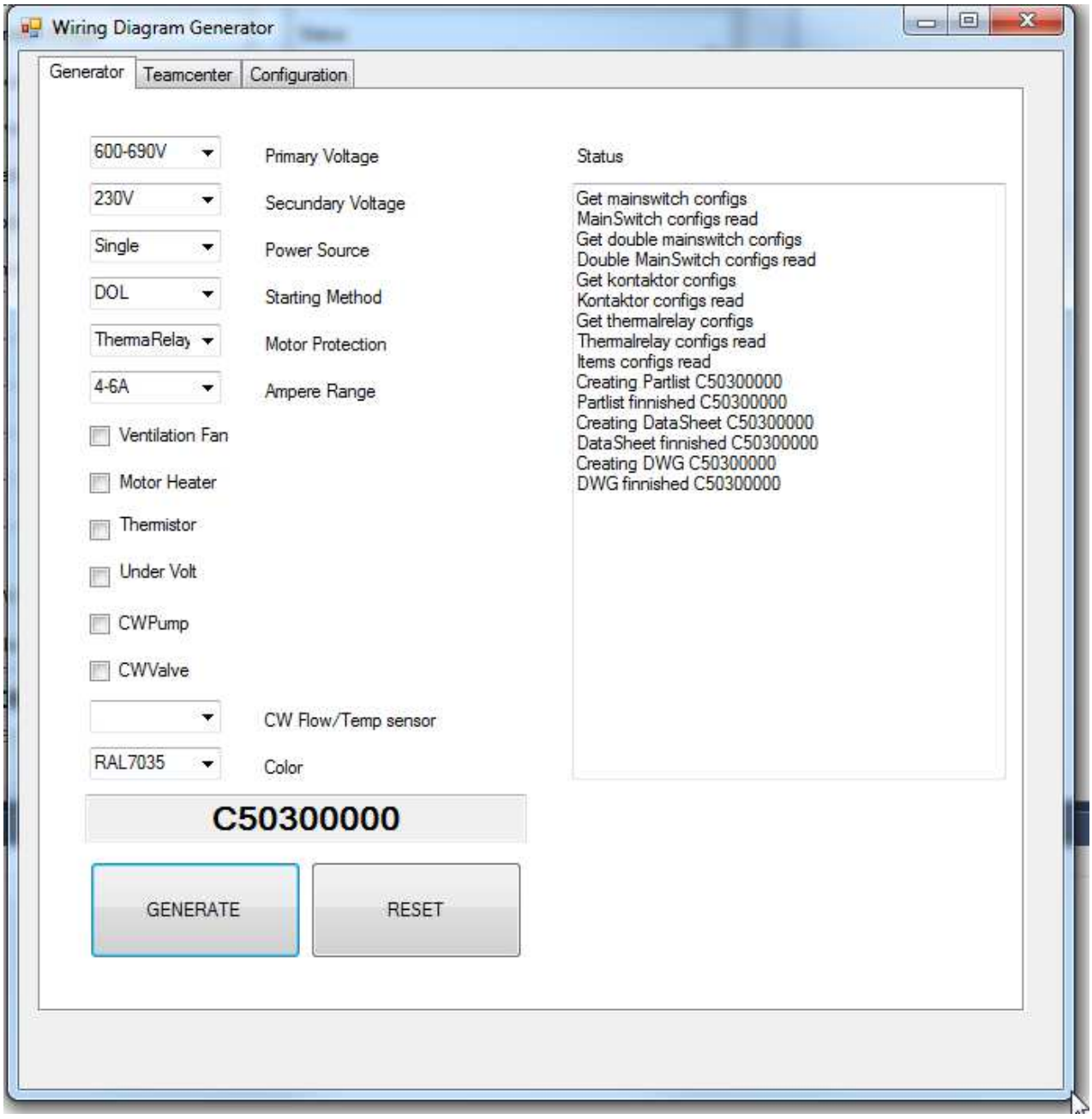
C.txt

replace	add
66110	66028
66115	66030
66120	66032
66125	66032
66130	66032
66135	66032
66140	66032
66170	66010
66175	66010

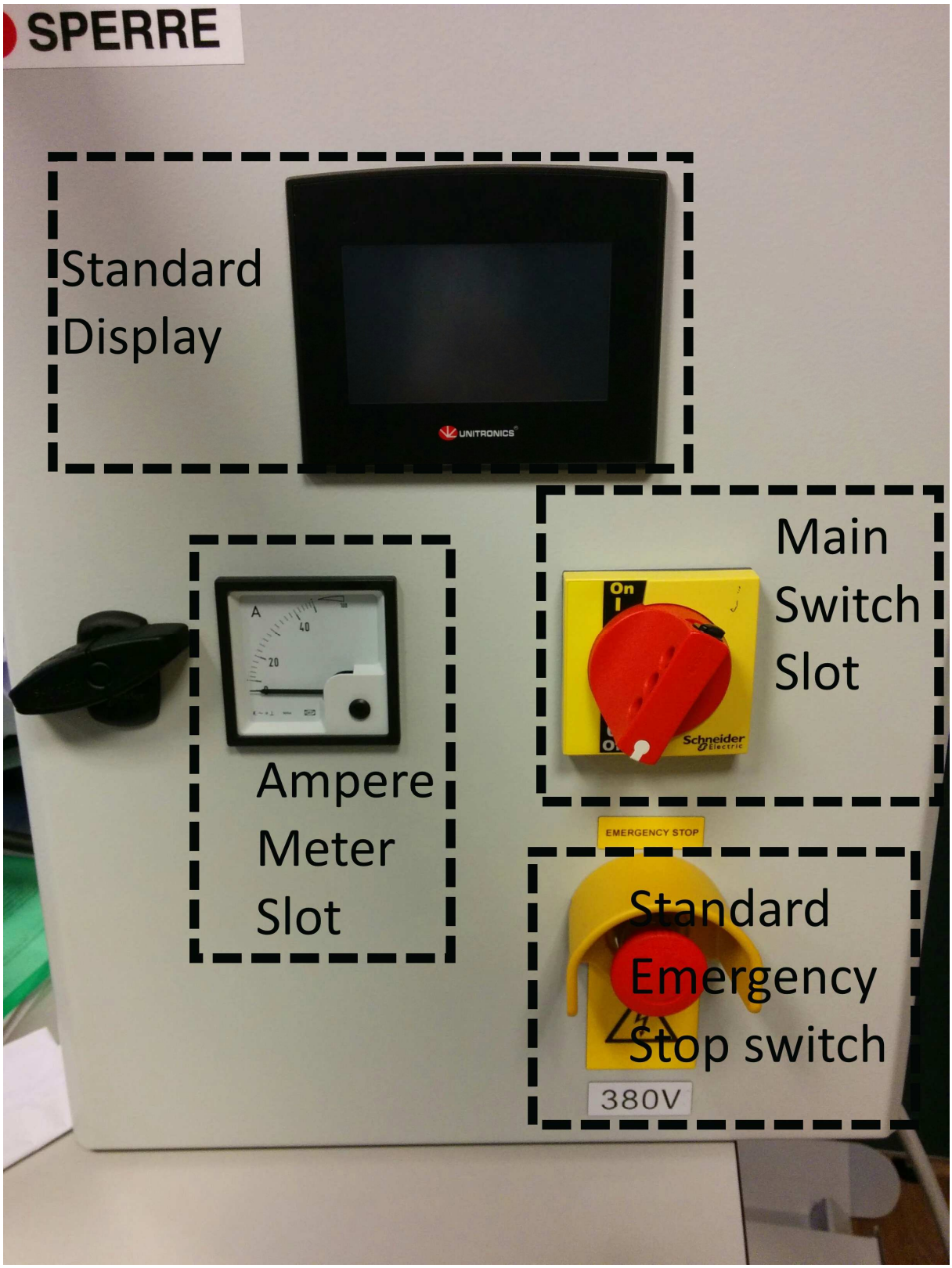
Combination table

ItemNr Char		Thermal relay Range [A]		Contactor [A]	Voltage Range [V]		Frequency Range[Hz]		Control Voltage [V]	Item A [AX]	Item B [3BX]
C[0-1]	C[3]										
C3	3	4	6	9	380	480	50	60	220	A1	3B1
C3	5	5.5	8	9	380	480	50	60	220	A1	3B2
C3	9	7	10	12	380	480	50	60	220	A1	3B3
C3	13	9	13	18	380	480	50	60	220	A1	3B4
C3	17	12	18	18	380	480	50	60	220	A1	3B5
C3	21	16	24	25	380	480	50	60	220	A1	3B6
C3	25	23	32	32	380	480	50	60	220	A1	3B7
C3	29	30	40	40	380	480	50	60	220	A2	3B8
C3	33	37	50	50	380	480	50	60	220	A2	3B9
C3	37	48	65	65	380	480	50	60	220	A2	3B10
C3	41	63	80	80	380	480	50	60	220	A3	3B11
C3	45	80	104	115	380	480	50	60	220	A3	3B12

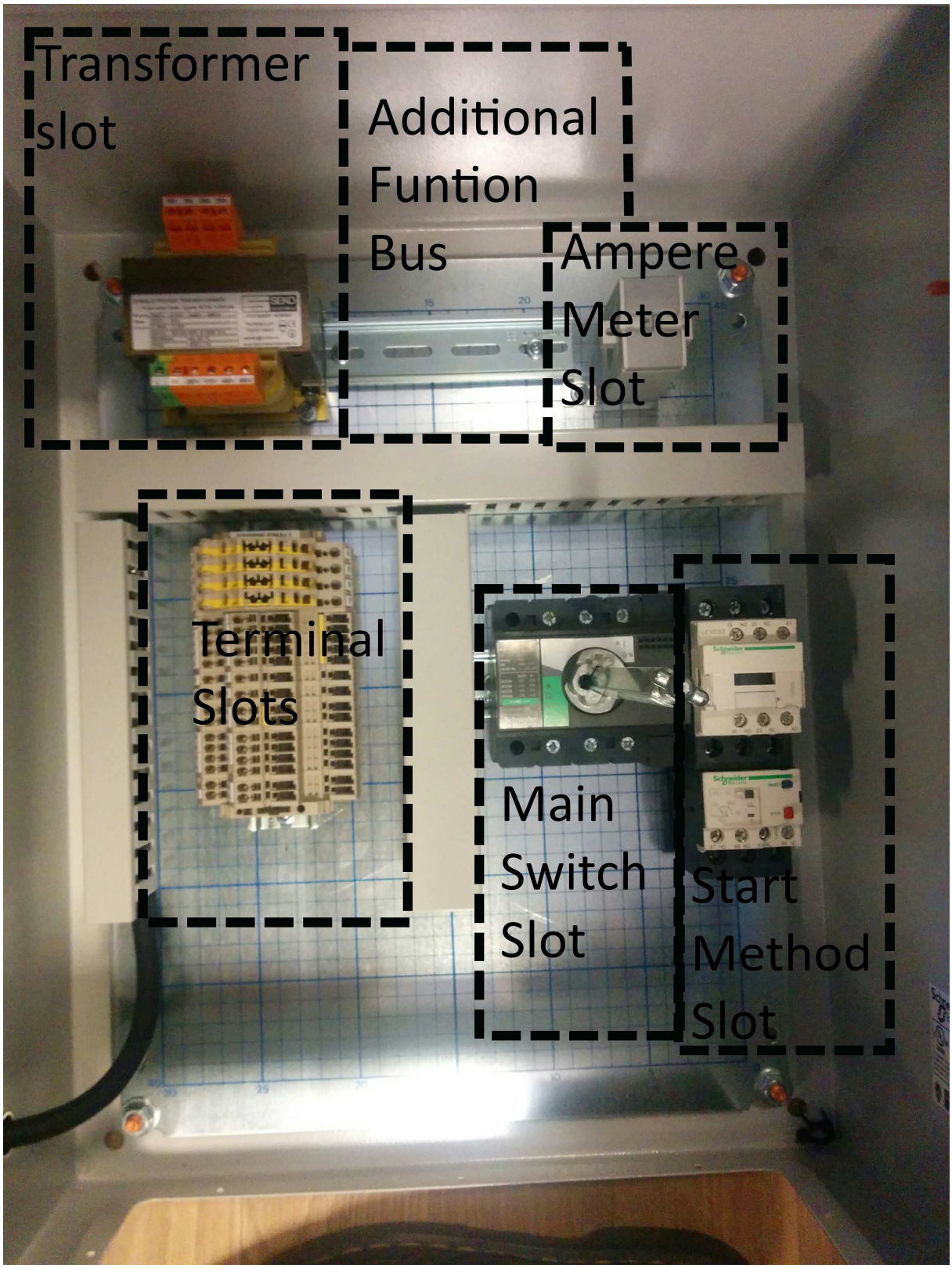
Documentation Generator



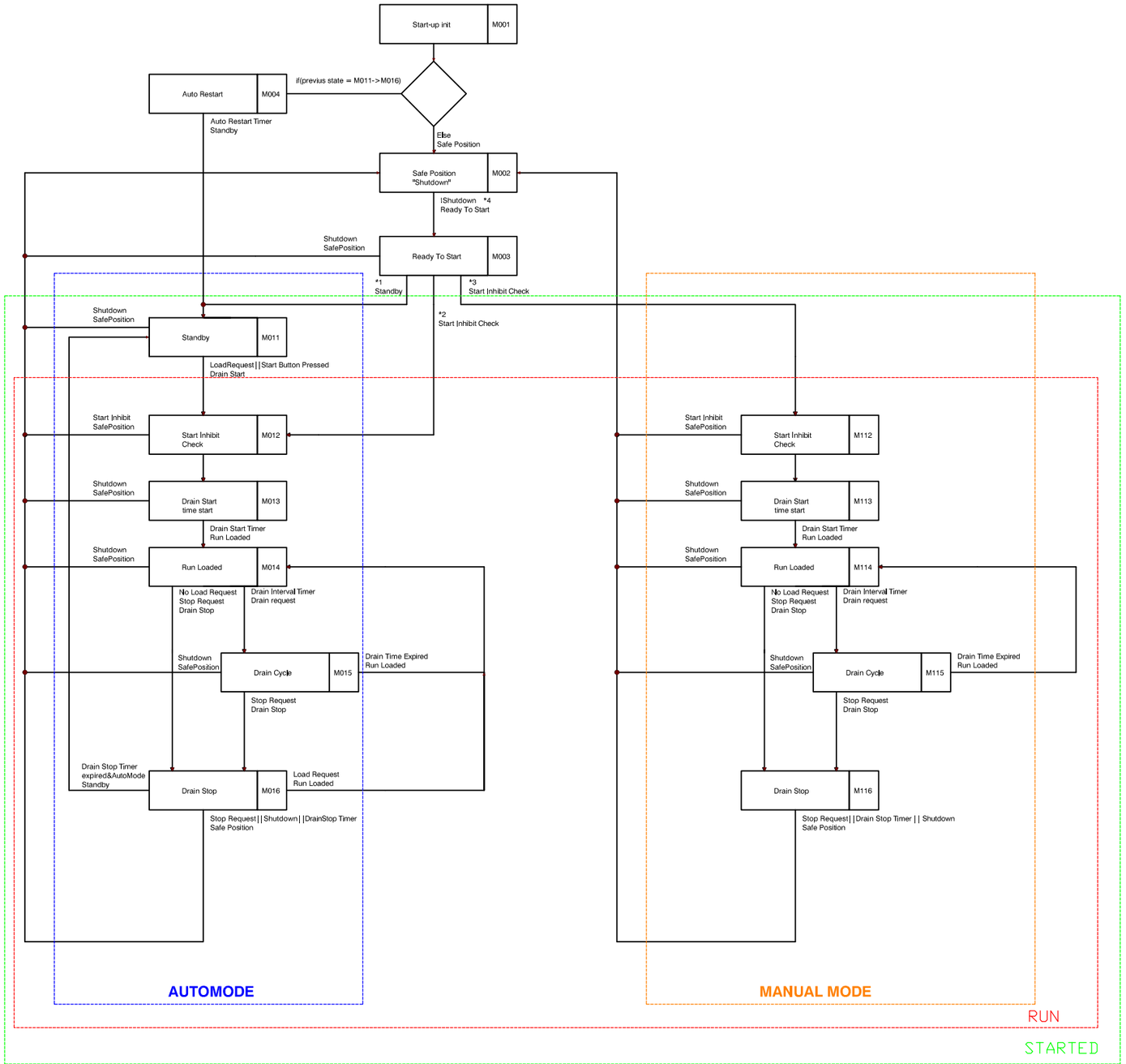
Starter layout front



Starter layout inside



0	1	2	3	4	5	6	7	8	9	10	11



Logic: & = AND, ||= OR, ! = NOT

1* Start CMD&Automode&Air Pressure>UnloadValue
2* Start CMD&Air Pressure<UnloadValue
3* StartCMD&ManualMode&Air Pressure<UnloadValue
*StartCMD
4*Shutdown activated by having an shutdown condition.
Shutdown is reseted by not having an shutdown condition and reset button is pressed.

Variables:
Load Request, is generated either from digital input or PT11.
PT11 is the pressure transmitter on the Air receiver.
Load Request (PT11)= Air Pressure <= Load Pressure
Load Request (DI Remote Load) = 1

Load Pressure and Unload Pressure is either set directly in set points or by Lead/Follow or Sea/Harbor mode.
Load Pressure < (Unload pressure - 1 Bar)

No Load Request (PT11) = Air Pressure >= Unload Pressure
No Load Request (DI Remote Load) = 0

StartCMD is generated either from Startbutton(local) or Remote digital input = 1 (remote mode).
StopCMD is generated from Stopbutton(local) or Remote digital input = 0

Timers:
Auto Restart Timer (15-120 Sec) set by LVL 2 user
Drain Start Timer (2-10Sec) set by LVL 2 user
Drain Interval Timers (30 min) set by LVL 2 user
Drain Time Open (1-2 Sec) set by LVL 2 user
Drain Stop Timer (2-10Sec) set by LVL 2 user

Counters
Running Hours (Counts time when compressor is in "Run".
Service Hours (Counts down to next service)
Service A repeats every 1000Hours,
Service B repeats every 5000Hours
Service C repeats every 10000 Hours

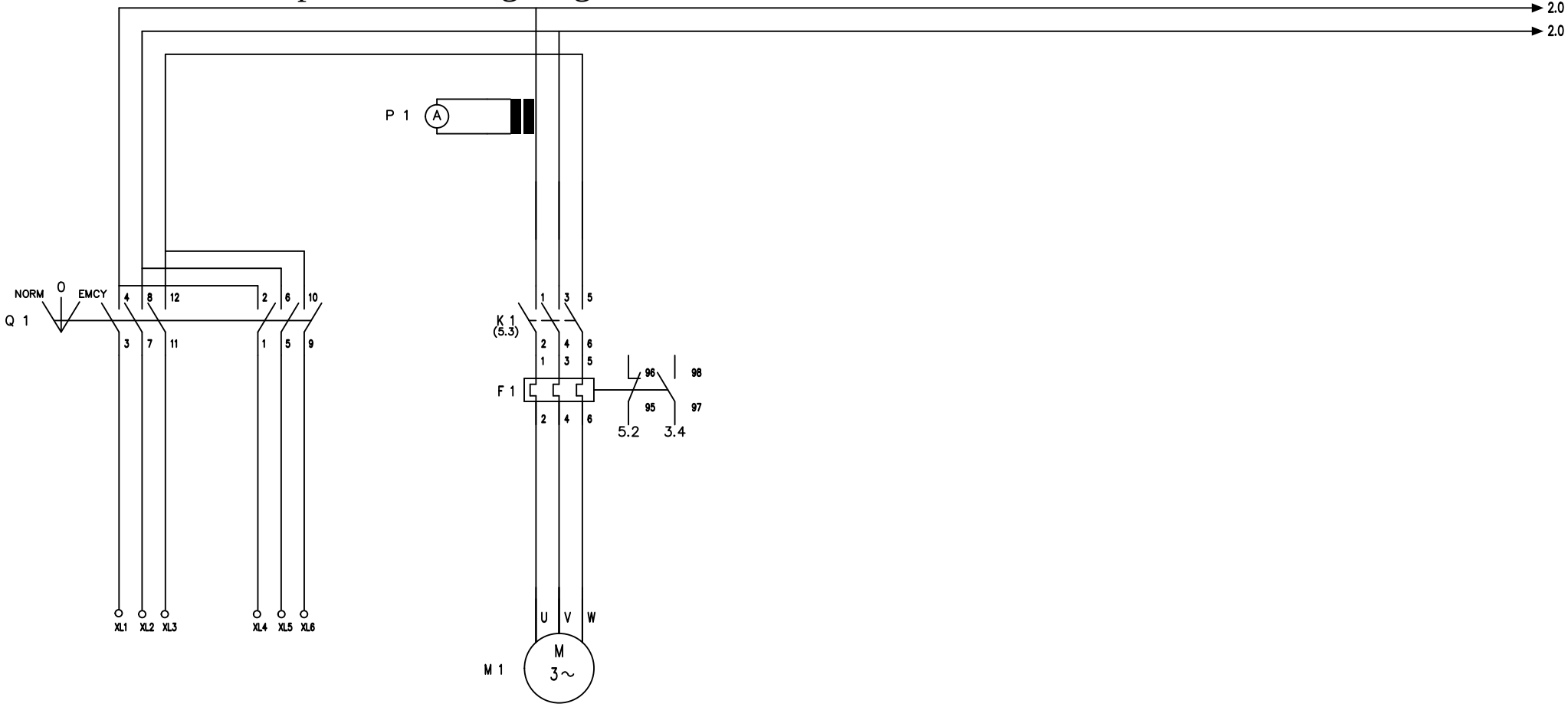
Status line on HMI display Statenumber + State name

SPERRE
INDUSTRI A/S
ADDRESS: 6057 ELLINGSØY – NORWAY
PHONE: 47 70 16 11 00 FAX: 47 70 16 11 10

	DATE	LAST REV.	SIGN		DRAWING No.:	
DRAWN	30.06.2015	22.03.2016	CAD	MACHINE STATE DIAGRAM		
APPROVED	30.06.2015	22.03.2016	JCB		PAGE 1	OF PAGES 1

	SUPPLY	SUPPLY		COMPRESSOR							

Documentation printout wiring diagram



SPERRE INDUSTRI A/S ADDRESS: 6057 ELLINGSØY – NORWAY PHONE: 47 70 16 11 00 FAX: 47 70 16 11 10	MAIN & CONTROL POWER		DATE	LAST REV.	SIGN	WIRING DIAGRAM
		DRAWN	06.10.2015		FAS	DRAWING No.: C32976001
		APPROVED				PAGE 1 / OF PAGES 7

Documentation printout part list

Control Cabinet Partlist

Sperre Industri AS

Type: C32976001

Voltage: 380-480 VAC
Current: 30-40 A

Ref	Description	PartNr
A1		123456789
S1	Emergency run switch	67050
K1	Contactor 40A 115V	66241
F1	Thermal relay 30-40A	66322
P1	Amperemeter 0-50A	8050-02
T1	Current transformer 50/5A	805-050
T2	Transformer 220VA 34/115/24	67124
Q1	Main switch dual supply 40A	66028
F4		123456
F6		123457
F8		123458
F8	Termostat (0-60C?) S:45C?	67905
M4	Ventilation fan for starters	67920
M4*	Ventilation filter and grate	67940
F9	Termistor relay	4175949
F10	Low voltage relay	66620

Sperre Industri AS
Tel +47 70 16 11 00
Fax +47 70 16 11 10
E-mail:industri@sperre.com

Sperre Rotterdam BV
Tel +31 103030342
E-mail: rotterdam@sperre.com

Sperre Asia PTE LTD
Tel +65 6763 63 00
Fax +656763 18 11
E-mail: asia@sperre.com

Sperre China
Tel +86 21 6507 0436
Fax +86 21 6507 0402
E-Mail: china@sperre.com

Documentation printout datasheet

Control Cabinet DataSheet

Sperre Industri AS

Manufactorer data	
Manufacturer	Sperre Industri AS
Compressor Type	Classic Range
Type	DOL Starter
Part Nr	C32976001
Version	
Supply	
Main Voltage	380-480 V
Frequency	50-60 Hz
Circuit Breaker	380A Slow 480 kA ICU
Technical Data	
Rating	IP54
Size	mm Height Weight Depth
Weight	21 kg

Sperre Industri AS
Tel +47 70 16 11 00
Fax +47 70 16 11 10
E-mail:industri@sperre.com

Sperre Rotterdam BV
Tel +31 103030342
E-mail: rotterdam@sperre.com

Sperre Asia PTE LTD
Tel +65 6763 63 00
Fax +656763 18 11
E-mail: asia@sperre.com

Sperre China
Tel +86 21 6507 0436
Fax +86 21 6507 0402
E-Mail: china@sperre.com

NTNU

Research Paper 2016

FOR

STUD.TECHN. Kai Brudevoll

Product Architecture with Automated Documentation Creation

Abstract

Products gets more and more customized for each customer. This leads to higher and higher varity in products. In this paper I have implemented product architecture with automated product documentation creation.

Product Architecture handling varity

Acronym list

BOM Bill of material

DOL Direct on line

Introduction

I choose to write about product architecture since in my company we have had a growing problem with our product architecture. The variations of product are so high that we are very seldom producing more than two equal products at a time. The numbers of product produced are very close to the number of different product produced. In addition we have a growing problem with starter designs that are out of date, but still reside in the system. The product in question is the electrical starter and control system for air compressors.

Literature

According to “The power of products platforms” we find the definition of a product family. A product family is a set of individual products that share common technology and address a related set of market applications. But what defines good product architecture? God metrics for good architecture is leverage. An architecture that has a low marginal cost to develop or incorporate new designs and variation into the architecture has good leverage.

Mark V. Martin has a good description of product variety. De dived variety into two axes: Spatial variation and generational variation. Spatial variation is defined as the product variety offered to the market at a given time, while generational variation is the variation over time.

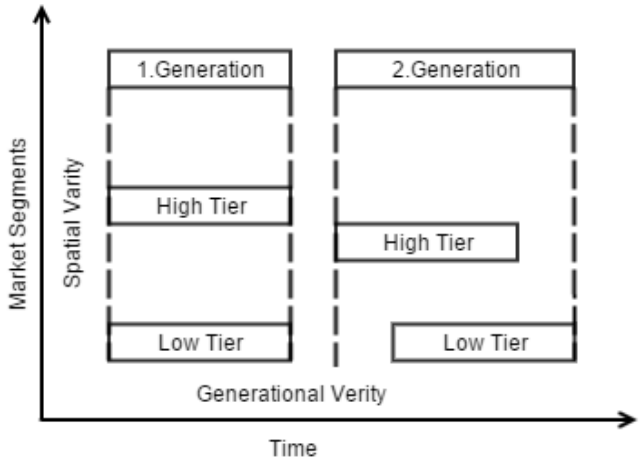


Figure 1 Spatial and generational variety

This shows us that our product architecture not only need to think of today's variation but also tomorrows.

Spatial variation

For our electrical starters we have several spatial variations. To organize the problem I divide the starter into attributes. The attributes have primary attributes and secondary attributes. Primary attributes are generally attributes that affects the rest of the products while secondary attributes as a minimum effect on the rest of the product. We start by finding the primary attributes that decide which starter we choose to use. Those attributes are motor voltage, current and starting method.

The first axis of variation is voltage. There are 3 normal voltages under 1000volts used for asynchronous motors in the world. 230, 380-480 and 600-690 Volt. This gives us the first 3 variants. The second attribute we chose to handle is starting method. We have two types of starting methods; Direct On Line (DOL) start and soft-start. This gives us 2 x 3 = 6 variants. The last attribute current comes in different ranges depending if soft-starting or DOL method is used. DOL has 10 different ranges and soft-starter has 5 different ranges. This gives us a total variant 3 x 10 + 5 x 3 = 45.

With these 3 attributes we have our base product. Now we can start adding options available to the customer. These are the secondary attributes that has a small effect on the rest of the product. Generally these options have to either affect documentation or the BOM to be considered. Heater, thermistor, remote start, munsell color, custom color and 110V control voltage. The

customer can select several options at the same time, except munsell and custom color they are mutually exclusive. This gives us a combination logic with 2x2x2x2x3x45 = 2160 possible spatial variants.

My solution to this problem is to handle the product as software. Manually creating 2160 variants of electrical drawings and part-lists takes a long time, but for a computer creating a product variant can take seconds. Generally a product needs some CAD diagrams, datasheets, certificates and part-lists. My case I need electrical diagram, data-sheet and part-list.

Diagrams

To create electrical diagrams we start by making one electrical diagram. This diagram is the one diagram all other diagrams get created from. We make draws each option gets two layers. One layer with the option implemented and one layer where the option is not implemented. For example we have 3 layers for motor voltage [230, 380-480, 600-690]. Only one of them can be active. Most CAD software's have an API that can be interfaced with your software. My case I use Bricscad. It has .net API drivers for windows. When using the api you can start Bricscad in the background and open the electrical diagram and modify layers, version number and the name of the diagram, then export the diagram to pdf.

Part lists

We start by defining the parts. We make one generic class for parts. Parts that has attributes outside what is defined in the generic class gets their own class. One example would be the contactor, larger the contactor used the larger the motor we can start. Attributes can also affect the attributive of the part. Voltage, ambient temperature affects the rating of components. This information can be stored in the different items classes.

- Items classes:
    - Contactor
    - Thermal Relay
    - Softstarter
    - Main switches
    - Item

The next step is to define the base of the product range. For my product this is easiest divided into two sub groups: A and B. A group is all the parts found in common for all enclosures. B group I use for the starting method and ampere scaling over each voltage range. A combination list is used to combine A and B BOMs to a base BOM.

Group C is used for different options. Category C has two functions: Add or Replace. Add function simply appends a part to the part-list. Replace function search the existing part-list and replace the part. An option like 115V control voltage needs to change all 230V parts to 115V. This means that each option for C has a list of all items that gets replaced and with what item it gets replaced with.

Bill Of Material				
A1		B2		C
Part 1	+	Part 21	+/-	Replace Part 21 with Part 31
Part 2		Part 22		

Table 1 Bill of material

When A and B is combined we get our base product. We give describe the product with the first 4 characters

Additional options are described with the next 5 characters. Each character represent 5 bits and each bit represent one option. If the starter should have a heater this would give character number 5 = 0 0 0 1 0 = 2. If we in addition to the heater wanted a thermistor we would get character 5 = 0 1 0 1 0 = A. We convert bit information into a triacontakaidecimal system also known as base32hex. 5 options are available for each extra character length to the part-number system used.

If the option is represented we run a function with the same name. This function opens a list of items that are either added or replaced, and then it activates a layer in the electrical diagram. With add or replace we can modify the part list to any variant with different options selected. If the customer wants a thermistor to monitor the motor temperature, one thermistor relay is added to part list and electrical diagram. If the customer wants 110V control voltage we have a large list that replace all 230V parts with its equivalent 110V.

	Primary Atributes		Options																			
	Volt & Start Method	Ampere	Function 1-5					Function 6-10					Function 10-15					Function 15-20				
Partnumber	0-9	00-99	0-V					0-V					0-V					0-V				
Bit			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
			Function 1	Function 2	Function 3	Function 4	Function 5	Function 6	Function 7	Function 8	Function 9	Function 10	Function 11	Function 12	Function 13	Function 14	Function 15	Function 16	Function 17	Function 18	Function 19	Function 20

Figure 2 New part number system

Generational variety

Over time the products architecture changes, these changes we call generational variety. These changes come from various causes. Martin comes up with various internal and external drivers for product change. I have listed these in Table 2 Drivers for generational change.

External drivers
Changing performance needs (weight, ECM, short circuit performance etc..)
Changing environment conditions(temperature, humidity etc)
New functions (new market, new technology)
Reliability improvements
Changing Regulations or standards
Competitor introduction of improved product (Higher quality or lower price)
Obsolescence of parts
Internal drivers
Cost reduction
Reduce assembly time
Reduce component types
Simplifying logistic
Increase meta data
Use lower cost technology
Reduce serviceability requirements
Increase serviceability
Improve component manufacturing process

Table 2 Drivers for generational change

Product change

There are two types of product changes one that requires an update to all products or adding new options to the system.
If there is a change in regulation that affects all of our products we need to update all documentations. In a perfect world all the different BOMs and diagrams for each product are updated, but when you have 2160 different BOM you do not have the time to update each product. The normal way to tackle this is to update the product when it's sold. This works reasonable well over a shorter period of time, but keeping track gets harder overtime.

There are several free bits that can be used to describe new options. If we have 2160 variants then after one new option we double it to 4320 variants. By implementing the option into the program we only need to implement it once and not a potential 2160 times. The marginal cost to manually implement new function equals

Cm= Base*2^(n-1)

Automation of product updates

To create a product you not only need to make the documentation, but also store the documentation in a PLM system. Brad Walton writes in his master thesis of a possible solution. In the paper he describes how you can automate data import and revision management in teamcenter. Teamcenter is a product lifecycle management program where you can store your documentation. Teamcenter organize its content into Items, Item revisions, and datasets. Item is a container that stores the ItemName and ItemNumber. Inside Item, Items Revisions are found. Item Revision uses sequential numbering system from A to Z. Under Item Revision datasets are stored. Datasets can be any type of file.

You can connect to teamcenter by using https calls from teamcenter .net library. With this library you can search the database, check revisions, create new revisions and upload datasets. Combining this with the software for documentation generation you can have a automated update system. See Figure 3.

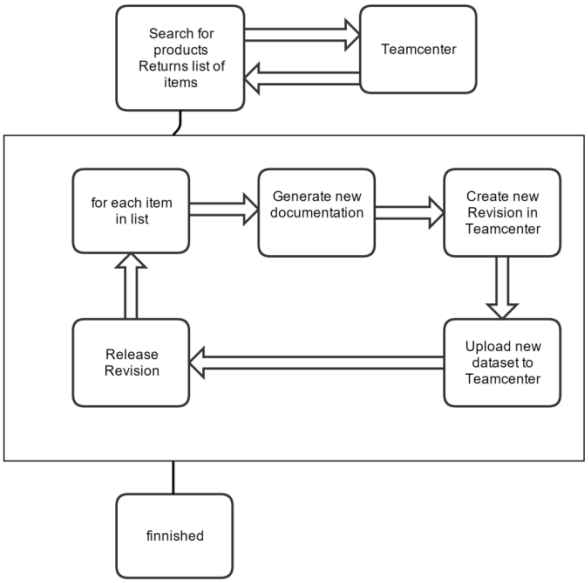


Figure 3 Teamcenter product update steps

The consequence of automation

By automate every step off the design of the product we see an interesting effect on the leverage the product architecture gets. Normally the cost to implement one more option is extensional, but with the automation we can argue its closer to constant. See Figure 4

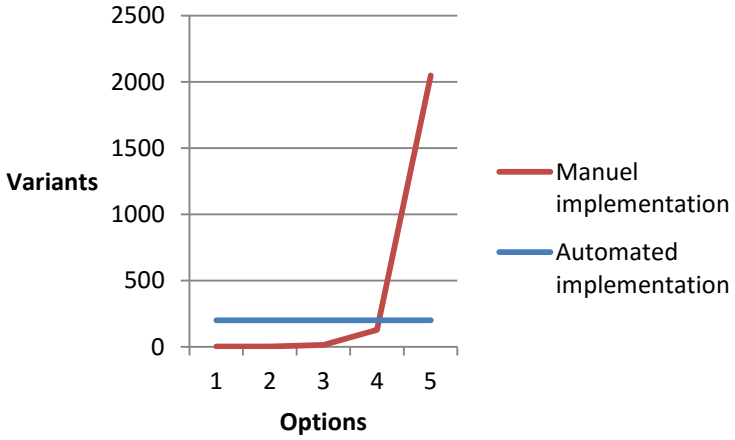


Figure 4 Marginal cost of implementing new options

The workload is higher when implementing the options for the first order, but once you have its compatible with all the options that has been created earlier. This would argue that implementing

an automated product generation would let the product architecture function for a longer period of time than an manual system.

Reference

[1] The Role of product architecture in the manufacturing firm – Karl Ulrich
Massachusetts Institute of technology, Sloan school of Management. Dec.1993
Research paper

[2] The power of product platforms - Marc H. Meyer & Alvin P. Lehnerd
The free press 1997.
Book

[3] Design for varity – Mark Valeton Martin
Department of mechanical engineering and committee on graduate studies of Stanford university in partial fulfillment of the requirements for the degree of doctor of philosophy Nov.1999
Dissertation

[4] Automated Data Import and Revision Management in a Product Lifecy –Brad Walton Brooks
Brigham Young University – Provo. Sept.2009
Master Thesis