3D Visualization and Explorer Tool for Different W31 Engine Configurations

Kim Enges

Bachelor´s thesis

Degree Programme in Mechanical and Production Engineering

Vaasa 2018

BACHELOR'S THESIS

Author: Kim Enges
Degree Programme: Mechanical engineering
Specialization: Mechanical Construction Systems
Supervisors: Franco Cavressi (Wärtsilä) and Kenneth Ehrström (Novia)

Title: 3D Visualization and Explorer Tool for Different W31 Engine Configurations

Date: February 23, 2018 Number of pages: 52 Appendices: 0

Abstract

This thesis was made for Design W31 at Wärtsilä Finland, Marine Solutions, Technology. Design W31 is a team of engineers, whose main tasks are to provide needed design for Wärtsilä 31 delivery projects as well as improvements and fixes of Wärtsilä 31 module variants.

The purpose of this thesis was to create the logics for positioning of engine components. The logics are intended for a 3D configurator, which will be used to create 3D engine assemblies for the W31 engine in Siemens NX. A 3D configurator can be useful in the engineer´s daily work. The engineer can enter the engine configuration data in the 3D configurator and get a 3D engine model in Siemens NX. By having this type of configurator to assemble and select the components needed for an engine assembly, the engineer can save a lot of time and avoid repetitive work.

The logics are created in a program called Rulestream, which is a client-server application from Siemens.

The result of the work is that approximately 80 percent of the logics for all Wärtsilä 31 modules are completed. The logics are stored in Rulestream. Rulestream and the logics for the Wärtsilä 31 module variants will in the future be integrated with the Internal Order Specification configurator (IOS configurator). The IOS configurator will provide the user interface for the 3D configurator.

Language: English Key words: 3D configurator, Siemens NX, 3D engine model

EXAMENSARBETE

Författare: Kim Enges

Utbildning och ort: Maskin- och produktionsteknik, Vasa

Inriktningsalternativ: Maskinkonstruktion

Handledare: Franco Cavressi (Wärtsilä) och Kenneth Ehrström (Novia)

Titel: 3D-visualiserings- och utforskningsverktyg för olika W31 motorkonfigurationer

---

Datum: 23.2.2018 Sidantal: 52 Bilagor: 0

Abstrakt

Detta examensarbete gjordes för Design W31 hos Wärtsilä Finland, Marine Solutions, Technology. Design W31 är ett team av ingenjörer, vars huvudsakliga uppgifter består av att tillhandahålla nödvändig design för Wärtsilä 31 leveransprojekt, förbättringar och korrigeringar av Wärtsilä 31 modulvarianter.

Syftet med detta examensarbete var att skapa logik för positionering av motorkomponenter. Logikerna är ämnade för en 3D-konfigurator, som skall användas för att plocka ihop motorsammanställningar i 3D. En 3D-konfigurator kan vara användbar i ingenjörens dagliga arbete. Ingenjören kan ange motorkonfigurationsdata i 3D-konfiguratorn och få en 3D-motormodell i Siemens NX. Genom att ha denna typ av konfigurator för att sammanställa och välja de komponenter som behövs till en motorsammanställning, kan ingenjören spara mycket tid och undvika repetitivt arbete.

Logiken skapas i ett program som heter Rulestream. Programmet är en klientserverapplikation från Siemens.

Resultatet av arbetet är att cirka 80% av logiken för alla Wärtsilä 31 moduler är klar. Logiken finns lagrad i Rulestream. Rulestream och logiken för Wärtsilä 31 modulvarianterna kommer i framtiden att integreras med Internal Order Specification-konfiguratorn (IOS-konfigurator). IOS-konfiguratorn kommer utgöra användargränssnittet för 3D-konfiguratorn.

Språk: engelska Nyckelord: 3D-konfigurator, Siemens NX, 3D-motormodell

---

OPINNÄYTETYÖ

Tekijä: Kim Enges
Koulutus ja paikkakunta: Kone- ja tuotantotekniikka, Vaasa
Suuntautumisvaihtoehto: Konesuunnittelu
Ohjaajat: Franco Cavressi (Wärtsilä) ja Kenneth Ehrström (Novia)

Nimike: W31-moottorin eri konfiguraatioiden 3D visualisointi- ja etsintätyökalu

Päivämäärä: 23.2.2018 Sivumäärä: 52 Liitteet: 0

Tiivistelmä

Tämä opinnäytetyö tehtiin Wärtsilä Finland, Marine Solutions, Technology Design W31:lle. Design W31 on insinööriryhmä, jonka päätehtävä on tarjota Wärtsilä 31-toimitusprojektien, parannuksien ja Wärtsilä 31-moduulimuunnoksien tarvittaavaa suunnittelua.

Tämän opinnäytetyön tarkoituksena oli luoda logiikka moottorikomponenttien asemoinnille. Logiikat on tarkoitettu 3D-konfiguraattoriin, jota käytetään W31-moottorin 3D-moottorikokoonpanojen luomiseen Siemens NX:ssä. 3D-konfiguraattori voi olla hyödyllinen insinöörin päivittäisessä työssä. Insinööri voi syöttää moottorin asetustiedot 3D-konfiguraattoriin ja saada 3D-moottorimallin Siemens NX:ssä. Koska tällainen konfiguraattori pystyy kokoamaan ja valitsemaan moottorikokoonpanoon tarvittavat komponentit, insinööri voi säästää paljon aikaa ja välttää toistuvaa työtä.

Logiikka luodaan Rulestream-ohjelmassa, joka on asiakas-palvelin sovellus Siemensin.

Työn tuloksena on, että noin 80% kaikesta Wärtsilä 31 -moduulien logiikasta on valmis. Logiikka on tallennettu Rulestreamiin. Rulestream ja Wärtsilä 31-moduulimuunnelmien logiikat tullaan tulevaisuudessa integroimaan Internal Order Specification -konfiguraattoriin (IOS-konfiguraattori). IOS-konfiguraattori antaa 3D-konfiguraattorin käyttöliittymän.

Kieli: englanti Avainsanat: 3D-konfiguraattori, Siemens NX, 3D-moottorimalli

Table of contents

Abbreviations

CAD	Computer-Aided Design
MFD^{TM}	Modular Function Deployment^{TM}
QFD	Quality Function Deployment
MIM	Module Indication Matrix
DFM	Design for Manufacturability
DFA	Design for Assembly
ETO	Engineer to Order
KBE	Knowledge-Based Engineering
PCM	Product Control Model
RSA	The Rulestream Architect Client
RSE	The Rulestream Engineer Client
IOS	Internal Order Specification
EBoM	Engineering Bill of Materials
PLM	Product Lifecycle Management
DCV	Delivery Centre Vaasa
WLSA	Wärtsilä Land and Sea Academy

1 Introduction

This Bachelor´s thesis is made for Wärtsilä Finland, Marine solutions, Technology. The thesis was made while I was working at Wärtsilä as a summer trainee 2017 and continued after the summer trainee contract ended. The Bachelor´s thesis was offered to me after attending the summer trainee interview. The scope of the thesis was to create a tool for Wärtsilä´s engineers to use for example when designing new module variants for the new W31 engine. The tool gives engineers the possibility to automatically generate 3D full engine assemblies or only portions of them, depending on the case at hand, and those can be visualized and navigated in Siemens NX.

1.1 Wärtsilä

Wärtsilä is a company that is a global leader in advanced technologies and lifecycle solutions for the marine and energy markets. Wärtsilä has operations in over 200 locations in more than 70 countries. Net sales in 2016 totaled 4.8 billion EUR, with approximately 18,000 employees. The company brings efficient, environmental solutions and fuel flexibility to the demanding market we have today. Wärtsilä´s portfolio is very extensive, offering engines and generating sets, lifecycle solutions, solar, automation, thrusters, ship design, energy storage etc. Wärtsilä is divided into three businesses: Services, Energy solutions and Marine solutions.

1.1.1 Services

Services deals with after sales business. The unit has a wide offering of services, developed to meet the requirements of the customers. Examples of the wide range of expertise and services offers include engine services, training services, lifecycle solutions, propulsion services etc. The importance of renewables, gas as a fuel and the increase of decentralized power generation are important drivers in this industry. Customers are facing a demanding market, which forces them to pursue every opportunity of cost savings and increased competitiveness, while maximizing the uptime and availability.

1.1.2 Energy Solutions

Energy Solutions are a leading global systems integrator that offers environmentally friendly solutions. The flexible and efficient solutions portfolio provides high value to customers and enables the transition to a modern energy system. Energy Solution´s offering consists of flexible baseload power plants, multi-fuel solutions and energy system integration capabilities, to name a few ones. The market drivers for Energy Solutions are growing because of today’s increasing demands for renewable energies, reduction of the carbon footprint and replacement of fossil fuels in favor of natural gas. Other drivers are a generalized global trend of economic growth and an improved standard of living for the majority of the Earth population.

1.1.3 Marine Solutions

Marine Solutions is a leading provider of innovative products and integrated solutions in the marine and oil & gas industries. Marine Solutions strives to continuously develop and transform to meet its customers’ needs. The business unit has a very complete offering, covering market segments such as oil & gas, merchant, cruise & ferry, navy and special vessels. Market trends and drivers such as globalization and population growth support the growth of the business. Technological developments and new inventions create a demand for new business models. Furthermore, the growing average *per capita* consumption and the increasing middle class population, especially in Asia, are both phenomena that are boosting the demand in the cruise sector. (Wärtsilä, 2017a)

1.1.4 Design team W31

The department that I have been working for is a part of Marine Solutions, called Design Team W31. The Design Team W31 is the department that is the requestor of this thesis work. Their main tasks are the following:

- To provide needed design for W31 delivery projects
- Design support for production
- Design, improvement and fixes of W31 module variants
- Creation of “main dimensions” drawings

My supervisor Franco Cavressi is also the Design Manager of the Design Team W31 and in his function, he was able to define the requirements for this thesis, while at the same time provide the technical support.

My main tasks during the summer trainee contract consisted of 2D, and 3D CAD work, using a Siemens provided software suite composed by Teamcenter and NX. Teamcenter enables data creation, management and storage, while NX is the actual CAD platform. Besides, of the 2D and 3D CAD work, I also have been working on this Bachelor´s thesis.

1.2 Problem definition

The problem is that when the engineers are designing new module variants, they have to be careful that the new variants do not interfere with other already existing ones. With a portfolio of hundreds of different module variants that can be combined in several ways in order to yield many different engine configurations, it becomes difficult to design and locate new parts on the engine. While doing this and at the same time, being certain that such parts will not collide with any of the existing ones, makes it even more challenging.

Building the 3D engine assemblies is also a rather time consuming activity, which is characterized by much repetitive work. Further to this, it is a manual activity and therefor the risk for human errors is present across the whole process. For example, when engineers are designing new module variants, they would benefit from visualizing the “over constrained” situation of the area affected by their design. Over constrained means, that all the different variants of a module are loaded and positioned correctly in the same assembly. Taking into account the space occupancy of every possible component in the surroundings allows the engineers to visualize the actual space that will for sure be available for the new design.

1.3 Purpose

The purpose of this work is to provide a tool in the form of a 3D configurator, that engineers can utilize in their daily work. Users of it can for example visualize various engine configurations. These can be delivery project specific or completely arbitrary (e.g. related to design studies for development projects). The generated 3D engine assemblies can then be utilized when working on new module variant designs, or for testing and investigating specific assembly combinations.

1.4 Aim

The aim is to create the logics for the 3D configurator. These logics are to be used in the 3D configurator for selecting and positioning the engine components.

1.5 Delimitations

My part of the project is limited to creating the logics. This was decided with my supervisor because of the project´s large scope. The thesis is a continuation of the Rulestream project, which was in the beginning phase at the start of the thesis.

1.6 Disposition

Chapter 2 describes theory around the W31 engine, modular design and Rulestream.

Chapter 3 explains the method used for the thesis.

Chapter 4 presents the results.

Chapter 5 gives a summary of the work.

2 Theory

This chapter gives a brief introduction of the W31 engine and some of the aspects that make it an innovative product that also enable the present work. After that, the concept of modular design and knowledge-based engineering is explained. Finally, the basics of Rulestream functions and usage are explained to give a better overview of the work carried out in this Bachelor´s thesis.

2.1 The Wärtsilä 31 engine

The newest, most advanced and efficient engine in Wärtsilä´s portfolio is the W31 engine. The engine is a 4-stroke engine that offers the best fuel economy in its class. (Wärtsilä, 2015b)

The W31 engine is categorized as a medium speed engine, it is available in many different cylinder configurations and it comes with three alternative fuel options: diesel, dual-fuel (DF) and spark-ignited gas (SG). The available cylinder configurations are 8V, 10V, 12V, 14V and 16V. A 20V cylinder configuration is also available for the Energy Solutions market. The power output for the different cylinder configurations ranges from 4.2 to 9.8 MW, at 720 and 750 rpm. (Wärtsilä, 2017b)

The W31 engine is specially designed with focus on a broad range of ship types and applications. It is suitable for diesel-electric or propulsion as a main engine, or as an auxiliary engine. It is possible to optimize the engine for running at either constant speed or along a propeller curve. (Wärtsilä, 2017b)

The Wärtsilä W31 engine was awarded a Guinness World Record title, being the most efficient 4-stroke diesel engine in the world. This title was awarded on May 26, 2015. (Wärtsilä, 2015a) The engine's diesel fuel consumption is what makes it the titleholder, being only 165g/kWh. What makes it possible for the fuel efficiency to reach these remarkably low numbers is the application of new technologies. These technologies are a high-pressure fuel injection system, double-stage turbocharging and adjustable inlet and exhaust valves actuation, in combination with a state of the art, latest generation engine control system. (Wärtsilä, 2015b)

The common rail fuel injection system is developed specifically for the Wärtsilä 31 engine. The same fuel injection system with multiple injection capability, in both the diesel and the dual-fuel engines, facilitates future conversion needs, as no separate pilot fuel needs to be added. The common rail system provides benefits like smokeless operations at all loads.

Another key technology to achieve reduced fuel consumption is the turbocharger. It is a second generation, two-stage turbocharging system. It has a pressure capability above 10 bar, with the turbocharger efficiency above 75%, compared to the typical efficiency level for a single-stage turbocharger of around 65-70%.

To exploit the full potential of these technologies, adjustable valve actuation is needed. Using adjustable valve actuation ensures the correct air-fuel ratio in both steady state and transient operation (engine-operating conditions). The hydraulic valve actuation does not require any periodic adjustments of inlet and exhaust valve clearance during operation. This is done at the scheduled service intervals, every 1000-hour for the valves, where the actuation mechanism is checked and adjustments are made if needed. (Wärtsilä, 2015b)

2.2 Modular design

Due to the market becoming more and more competitive with low prices, limited resources and decreasing market shares, it becomes necessary to get more out of the available resources. A broad and high quality product portfolio along with lowered costs for production and development per produced unit is one way to increase a company´s market shares. One way to do this is to utilize resources to design customer specific products, by combining sub-functions into different product variants. To accomplish this, the company can divide the product into modules. This can have the following effects: (Johannesson, et al., 2013)

- Higher flexibility – fast and easy product changes can be made since they will only influence limited parts of the product.
- Reduced product development lead-time – when the interfaces between modules have been defined, it is possible to have parallel development activities.
- Shorter lead-time in production – parallel manufacturing of modules.
- Less risk-taking in new development – shortened lead times means less capital tied up in production.
- Improved quality in production – quality testing of modules before final assembly.
- Fewer item numbers to manage – leads to reduced material and purchase costs.
- Service and upgrading – easier due to standardized interfaces.
- Administration – efficient quoting, planning and designing. (Ericsson & Erixon, 1999)

Modular Function Deployment^{TM} (MFD^{TM}) is a structured method, used to find the optimal modular product design by considering the company´s specific needs. The method consists of five steps: (Ericsson & Erixon, 1999)

1. Defining customer requirements, using a Quality function deployment (QFD) analysis for monitoring competitive position and customer needs.
2. Sub function based generation of technical solutions taking into consideration current manufacturing criteria.
3. Identification of possible modules, using a Module Indication Matrix (MIM).
4. Evaluation of proposed modularization.
5. Detail construction of module using traditional DFM and DFA methods. (Johannesson, et al., 2013)

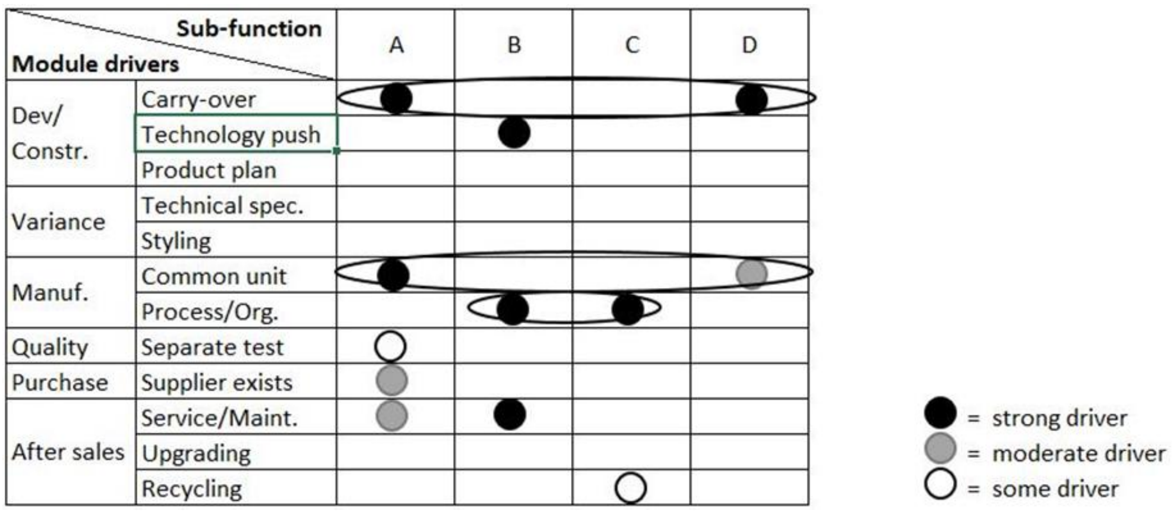
The Module indication Matrix, MIM, reminds of the QFD-matrix. In the MIM, the sub-function are in the columns, while the module drivers are in the rows.

Modularization is dividing a product into modules with set interfaces, driven by specific reasons. These specific reasons are the module drivers. Common module drivers in engineering companies are: (Johannesson, et al., 2013)

- **Development and construction**
  - *Carry-over* – designs that carries over to new product generations.
  - *Technology push* – the opposite of carry-over. A redesign of the part during the product life cycle to fit with new implemented technology.
  - *Planned development* – a product development plan is a strategic tool that describes product changes and specifies their launching dates.
- **Variance**
  - *Technical specification* – the company can change the technical specification of a product. The power unit of a printer for example. The printer is delivered with a different power unit, depending on the geographic market.
  - *Styling* – some product parts may be influenced by trends and fashion. Styling modules that typically contain visible parts should be used to underline product identity. Changing the color or design is an example of styling.
- **Manufacturing**
  - *Common unit* – functions that can have the same physical form in the products variants should be in common units.
  - *Process/Organization* – parts manufactured using the same manufacturing processes should be in common units to reduce lead-time.
- **Quality**
  - *Separate testable modules* – if the part has a function that is testable before the final assembly, it should be in a separate testable module.

- **Purchase**
    - *Supplier exists* – combination of parts and systems purchased from suppliers into common modules are preferred.
- **After sales**
    - *Service/maintenance* – quick service and maintenance in the field is often an important customer requirement. Combining parts that are exposed to service and maintenance into a service module makes the after sales part easier.
    - *Upgrading* – products that can be upgraded, rebuilt or have functions added in the future should be combined into a common module.
    - *Recycling* – parts with a limited number of different materials used in the product, enables a higher degree of recycling. These combined into specific modules simplifies disassembly before recycling or disposal.

An example of a MIM with the commonly used module drivers and sub-functions can be seen in Figure 1. In the matrix, the sub-functions A, B, C and D can then be against the module drivers. The marks represents a scale of how strong the module driver is for each sub-function. For example, sub-function A and D are both marked as strong carry-over and common unit. This is a reason to combine them into a common module. The purpose of the MIM is to examine how common module drivers affect the different sub-functions. Sub-functions can also have multiple module drivers as well as unique module drivers. It is preferred to keep these as a single module. (Johannesson, et al., 2013)



Module drivers		Sub-function A	B	C	D
Dev/ Constr.	Carry-over	●			●
	Technology push		●		
	Product plan				
Variance	Technical spec.				
	Styling				
Manuf.	Common unit	●			◍
	Process/Org.		●	●	
Quality	Separate test	○			
Purchase	Supplier exists	◍			
After sales	Service/Maint.	◍	●		
	Upgrading				
	Recycling			○	

● = strong driver
◍ = moderate driver
○ = some driver

Figure 1: Module-Indication-Matrix, MIM. **(Johannesson, et al., 2013)**

The Wärtsilä W31 engine is the first engine in Wärtsilä´s engine portfolio using a modularized product structure. With the modular design of the W31, maintenance work becomes easier and more efficient, which leads to maximized uptime. The modular design concept shifts from using single parts, to instead using exchange units, with standardized component interfaces. This enables the engine conversions to be faster and more efficient because there is no machining required, for example when converting an engine to use a different fuel (diesel to gas). (Wärtsilä, 2017b)

The modular design approach splits a system into smaller parts, also known as modules. These modules can then be used independently in different systems, as they use a standardized component interface. A few examples where modular design is used are vehicles, machines, architecture, programming etc. Examples of modular systems, containing module variants can be cars, engines, tools, process systems and many more.

A very simple example of modular design is a vehicle. The vehicle can have a large catalog with parts that can be added or removed without affecting other parts of the vehicle, e.g. upgrading to a more efficient engine does not require any changes to be made to other part of the vehicle.

Another example is the computer, which is one of the best examples of modular design. The computer include modules like processors, graphic cards, hard drives etc. These parts can be replaced with different parts, as long as the new part supports the standard interface for the computer. (Wikipedia, 2017b)

2.3 Rulestream

Rulestream is a client-server application from Siemens. It was built with the goal to enable efficient Engineer to Order (ETO) applications. (Siemens PLM, 2017b)

The Rulestream software for Engineer to Order aids companies that meets challenges, by allowing them to quickly engineer product to unique customer specifications, and in that way improve sales and order processes. (Siemens PLM, 2017a)

Engineer to Order (ETO) is a technique used to improve profits for companies that provides unique tailored solutions to customers. ETO begins with selling a product that does not have a set design. The product is expected to result in a new unique product, e.g. a project specific engine configuration. (Wikipedia, 2017a)

The Rulestream software is a descendant of the Knowledge-Based Engineering industry (KBE). Rulestream has simplified the process of Knowledge-Based Engineering by providing a streamlined, graphical interface that protects the experts from most of the complexity of object-based and demand-driven programming. This is done by eliminating considerable obstacles and challenges in completing KBE applications by:

- Enabling experts to enter and maintain rules.
- Allowing continuous rule updates to keep up with the business.
- Providing a rule execution platform to speed implementations and focus on rules.
- Making knowledge searchable and publishable to maximize value of captured rules. (Siemens PLM, 2017b)

Rulestream understands constructs like parts, assemblies, dimensions, 3D model features, drawings, standards parts list etc. The software understands part modeling along with the steps that need to be completed to take a set of requirements to a documented and released design. All of this makes the software a very efficient tool for companies that engineer products to customer specifications. It is also efficient for Companies that offers standard products since captured standards can be automatically applied to new designs or decisions. (Siemens PLM, 2017b)

2.3.1 Rulestream terminology

Application, a container in the Rulestream Architect client for part families.

Part Family is the definiton of a type of part that holds a set of attributes and rules which define the part. By assigning or calculating values to the attributes, an **Instance** can be created based on the definition of the part family.

Product Control Model (PCM), the design structure of part families in the Rulestream Architect client. It defines the product that is engineered or manufactured to order. The tree is often a hierarchical tree of part families but it can also be a non-hierarchical tree.

Top-Level Part Family, usually the final design assembly that is being created. It is below the PCM.

Property, describes a feature or attribute of a part family. Properties can be calculated using rules or formulas; they can be retrieved from a component database; or they can be specified at runtime. The propery type may be **Manual** or **Formula driven**, meaning the value is either provided by user input only, or determined by a formula.

Specification, the general definition of the type of data the property holds and defnitions of where the knowledge came from.

Constraint, the rule of how the property gets its value and how that value is controlled by the end-user.

Process, a collection of user interface process steps, which contains user interface inputs and outputs.

Project, a logical grouping of one or more designs to be completed in the Rulestream Engineer client.

Line Item, used to create an instance of a finished product (PCM) in the Rulestream Engineer client. A project can have multiple line items.

Subscription, an agreement to receive a publication, an article, or a service. A subscription involves a source and a destination. In Rulesteam, a subcription is a linking mechanism.

Subpart Collection, is the parent/child relationship between partfamilies that signifies that one part family (the parent) has a part family (the child) as a part of its definition. This relatioship allows the part families to reference each other and promotes a hierarchical tree structure of part families.

A **Subpart** refers to the nth part family in a subpart collection, where n is a number between 1 and the quantity of part families in the subpart collection. The Subpart is also known as an Item in a subpart collection, while “n” is the Subpart ID.

The **Formula Builder** is a tool built into Rulestream Architect that provides a simple point-and-click interface for building complex formulas. The interface provides graphical navigation to elements in the model tree as well as Rulestream defined global functions, VB.NET functions, and a history of every rule that has been saved for the property. (Siemens PLM, 2017b)

2.3.2 Rulestream software

There are four major functional parts of a Rulestream system.

- **The Rulestream Architect client (RSA)** offers a point-and-click interface that allows the user to capture and manage rules and processes to a database.
- **The Rulestream Engineer client (RSE)** allows users to gather requirements for a specific project and then use the previously captured knowledge to automatically create designs such as 2D and 3D CAD models, reports and bills of materials (BOM).
- **The Rulestream Thin Client**, a web interface that allows a user on the internet to gather requirements and then use the same captured knowledge to automatically complete a design of a custom product and generate a quotation online.
- **Rulestream Platform Server** provides a database for the storage of the knowledge, design models, design instances and component information. This is where everything is stored. (Siemens PLM, 2017b)

2.3.3 Rulestream Platform Server

The Rulestream Platform Server is the server of the application. With the use of Microsoft´s SQL Server, it hosts three databases:

- The Rules Database, stores the captured knowledge of processes and rules. It also stores definitions of what requirements may be specified when beginning a new project.

- The Project Database stores actual requirements specific to each design, and the state of the design.

- The Component Database stores information on standard parts that are available for use within a design. It is optional, and is not Rulestream-specific (that is, it may be independently created in SQL Server).

2.3.4 Rulestream Architect

The Rulestream Architect client is used to develop applications. The client interacts with the Rulestream Platform server as shown in Figure 2. It provides the user an interface to capture a company´s product design-knowledge, establishing a live framework so that engineering rules are consistently applied. There is no need for complex software programming.

Rulestream can capture the product rules, different variations and the context dependencies of the components. Rulestream Architect automatically turns the inputs into an executable program, run within Rulestream Engineer, to produce for example a 3D engine assembly. (Siemens PLM, 2017b)

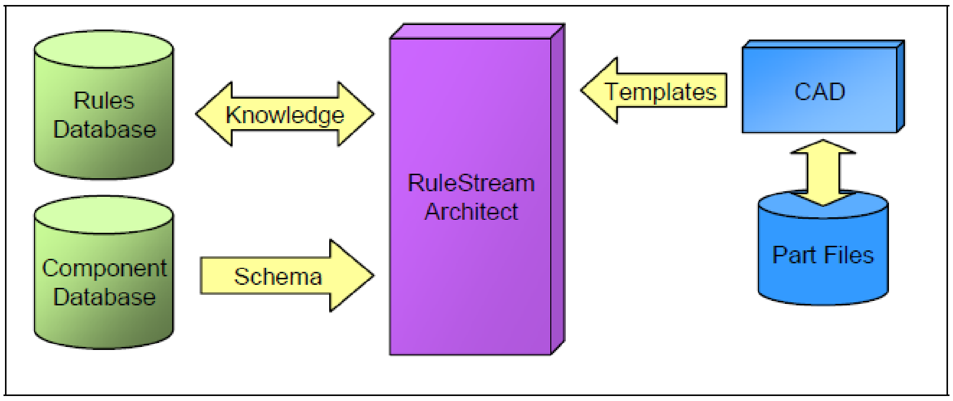


Figure 2: Rulestream Architect data flow. **(Siemens PLM, 2017b)**

2.3.5 Rulestream Engineer

The Rulestream Engineer client takes the information that has been gathered through Rulestream Architect and automatically stored within Rulestream Platform Server, to build one master product model, Product Control Model (PCM), which is composed of four default models: an input model, a design model, an analysis model and an output model. These are built automatically using rules that can be refined interactively by the users depending on the requirements. The Product Control Model can generate unique product designs, bill of materials, 2D drawings and 3D product geometry. See Figure 3 for an illustration of the data flow. (Siemens PLM, 2017b)

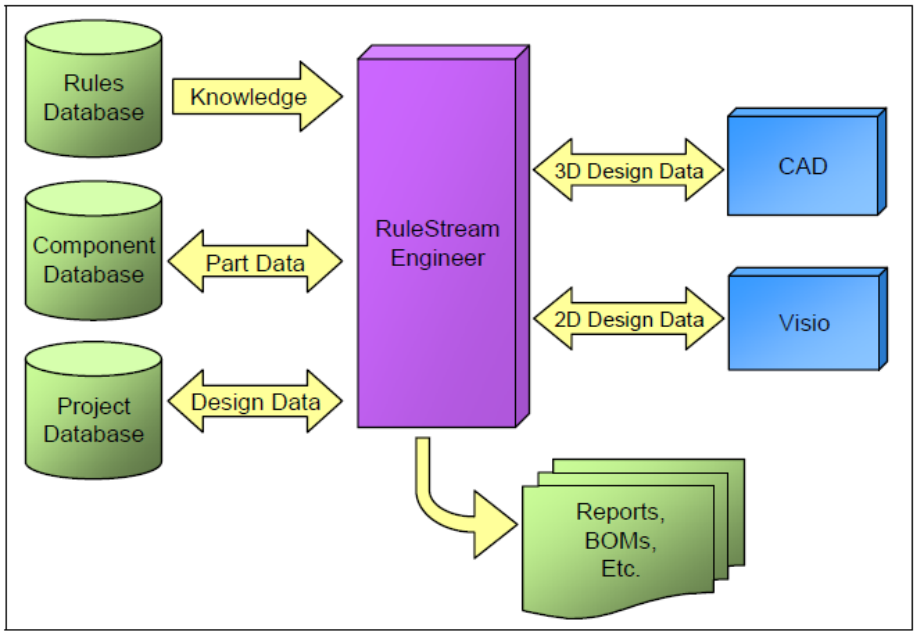


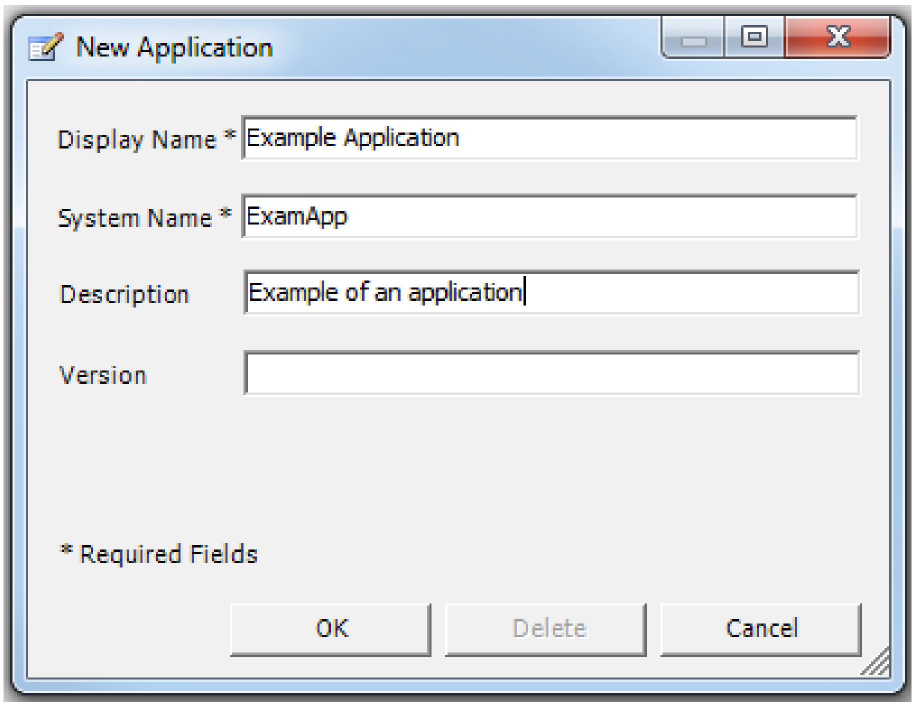
Figure 3: Rulestream Engineer data flow. **(Siemens PLM, 2017b)**

2.4 Rulestream features

This sub-chapter outlines the features used when the configurator logics were created. Different examples are provided in order to simplify the explanations.

2.4.1 Applications

The user can create an application in Rulestream Architect. The application is the owner of one or many part families. An example application is created in to explain the different features below, see Figure 4. (Siemens PLM, 2017b)


New Application

Display Name * Example Application

System Name * ExamApp

Description Example of an application

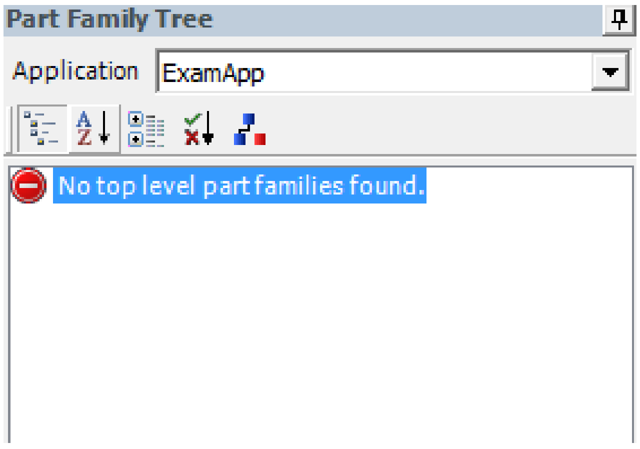
Version

* Required Fields

Figure 4: Example application in Rulestream Architect

2.4.2 Part family tree

The part families created are organized into a tree within this specific application. This is called the product control model. The application currently owns no part families in Figure 5. (Siemens PLM, 2017b)


Part Family Tree

Application ExamApp

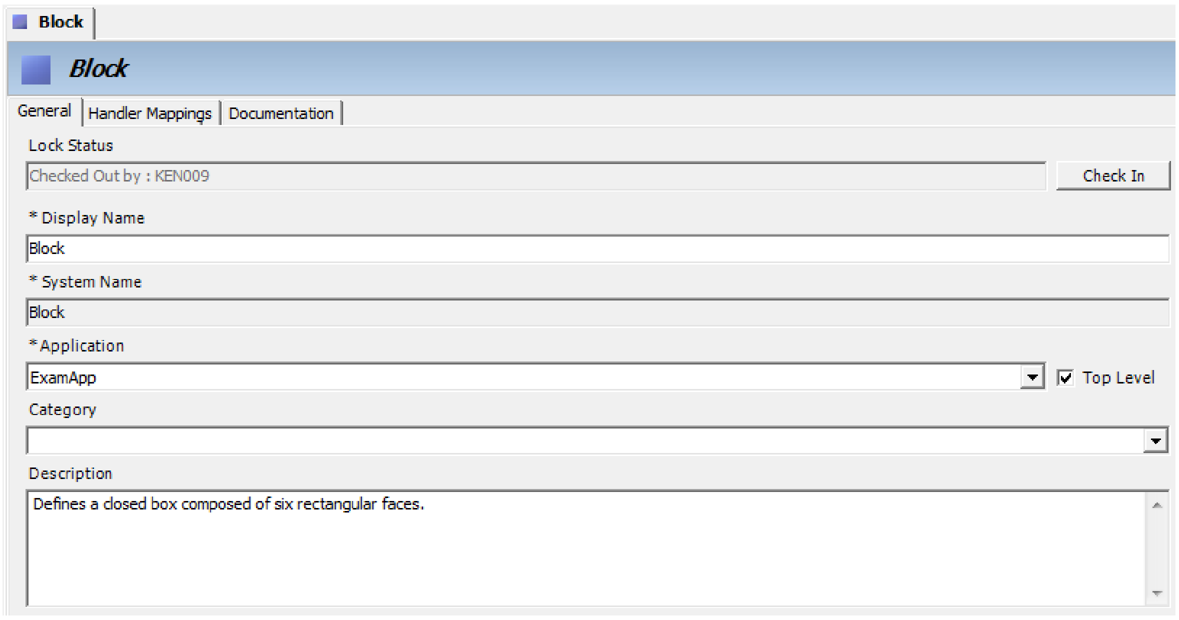
No top level part families found.

Figure 5: Empty part family tree in the Example application

2.4.3 Part families

A part family is a building block of a product model. The user can attach model definitions and rules to the different part families, making it into a type of “recipe”.

In Figure 6, a block is created as a part family for this example. In this case, the block will be the only part family created.



Block

Block

General | Handler Mappings | Documentation

Lock Status

Checked Out by : KEN009

Check In

* Display Name

Block

* System Name

Block

*Application

ExamApp

[✓ Top Level]

Category

Description

Defines a closed box composed of six rectangular faces.

Figure 6: Creating a block as a part family

There are several fields to fill when creating a part family. The required ones are marked with a star (*), see Figure 6. Explanation of the fields:

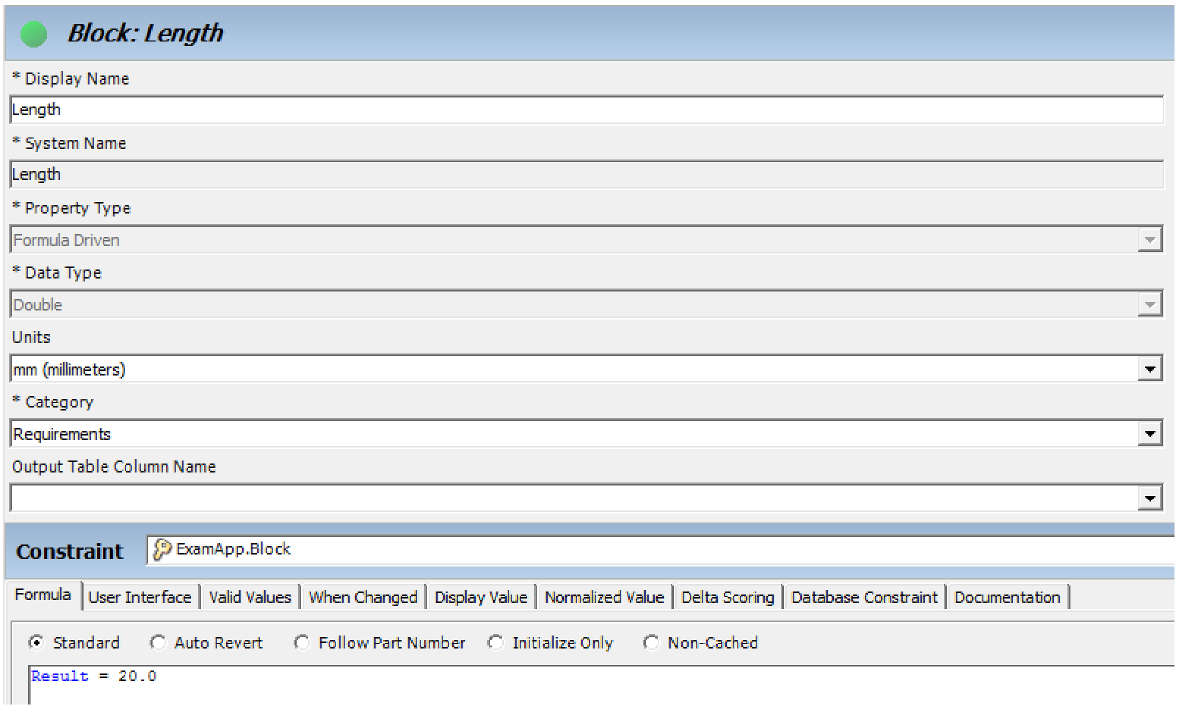
- Display name – This is the name displayed to the end-user. It has to be unique across all applications.
- System name – This field fills automatically by the display name, without spaces and special characters. The system name is the name used in the rule writing.
- Application – Identifies which applications owns the part family. The user can assign applications that are not the current application. In this case, the part family to the example application is assigned.
- Top part – Checking the top-level checkbox designates the part family will be a top-level part family. The top-level part family is then the entry point in the run-time model.
- Category – This field is used to group part families. This is left empty.

- Description – The user can describe the part family with information that is viewable at run-time. (Siemens PLM, 2017b)

2.4.4 Properties

The user can define properties, which describes a feature or attribute of the part families. A property can be a geometric characteristic such as length or height of a part, or a physical characteristic such as volume. Properties can also be a constant; they can be calculated by using a rule or formula; or they can be specified at run-time. The properties in Rulestream are composed of specifications and constraints. Specifications are general definitions data the properties holds and definitions of where the knowledge comes from. Constraints are the rules of how the properties receives their values and how the users control the values. (Siemens PLM, 2017b)

For the block part family created in chapter 2.3.4, four different properties are added. These are length, width, height and volume. For the width and height properties, the user in this example is restricted to a list of allowed values. The length property is not restricted. This is explained below in valid values. The volume property has the visible box checked and the enabled box unchecked. This is explained below in user interface.


Block: Length

* Display Name: Length
* System Name: Length
* Property Type: Formula Driven
* Data Type: Double
Units: mm (millimeters)
* Category: Requirements
Output Table Column Name:

Constraint: ExamApp.Block

[✓ Standard] Auto Revert Follow Part Number Initialize Only Non-Cached

```
Result = 20.0
```

Figure 7: Length property

Figure 7 shows the length property created. The property form divides into two regions: specifications and constraints.

In the **specification** region, the display name field and the system name field works the same way as described in chapter 2.3.4.

- Property type – The property is either manual or formula driven. If it is manual, then the user provides the input. Formula driven means that there is a formula that determines the value.
- Data type – The user can choose between a set of different data types, see Figure 8 below. Depending on which data type used, the property can only contain such data.

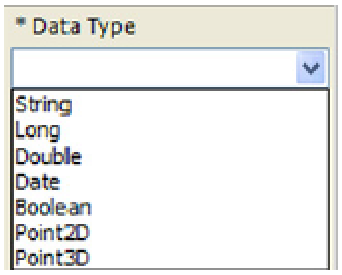


Figure 8: Data type selections

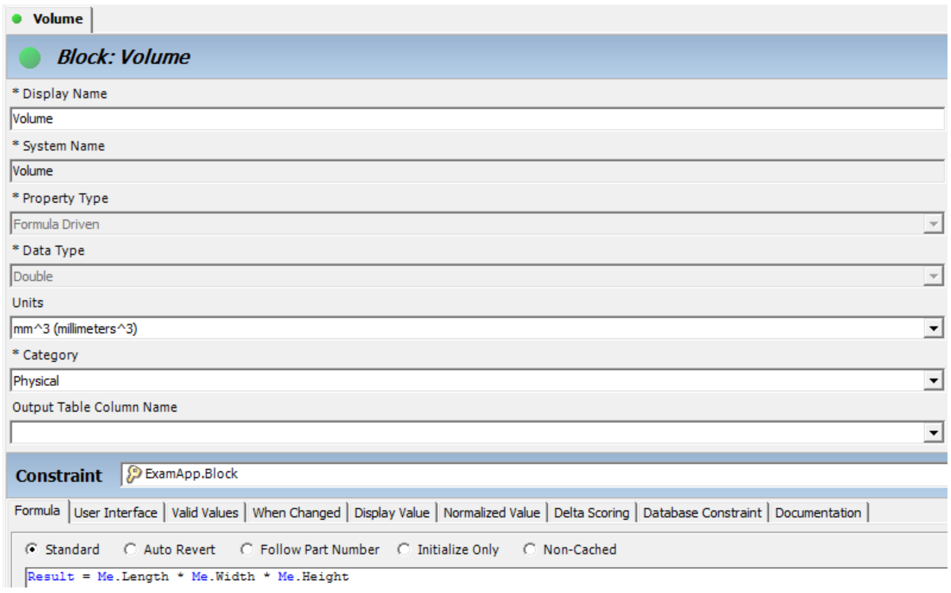
- Unit – A unit of measurement applicable for the property. Properties might not always require a unit, such as String data types.
- Categories – The dropdown list in the field allows the user to select in which category the property is grouped in the All specs field.

In the **constraint** region, the following fields are available for the user to constrain the property´s value:

- Formula – This is where the user writes the formula depending on the chosen data type. The formula is limited to 6500 characters. For the volume property, a formula to calculate the volume of the block is created, see Figure 9.
- User interface – The selection in this tab determines how the property behaves at run-time. Possible selections: **visible** and **enabled**. The user may also use the formula field to write advanced rules. By selecting the **visible** option the property becomes visible in the user interface, otherwise it is hidden. By selecting the **enabled** option,

the user has the possibility to override the default value manually. If not selected, the property will appear as read-only in the user interface.

- Valid values – This tab allows to user to create a list of choices, containing numerical or text values. Rulestream has a tool called Valid Values List Builder. See the example in Figure 10 and Figure 11 below. If the checkbox named "Restricted" in Figure 11 is checked, the user can only choose between the valid values. If left unchecked, then it is possible to override the values. (Siemens PLM, 2017b)


● Volume

Block: Volume

* Display Name: Volume

* System Name: Volume

* Property Type: Formula Driven

* Data Type: Double

Units: mm^3 (millimeters^3)

* Category: Physical

Output Table Column Name:

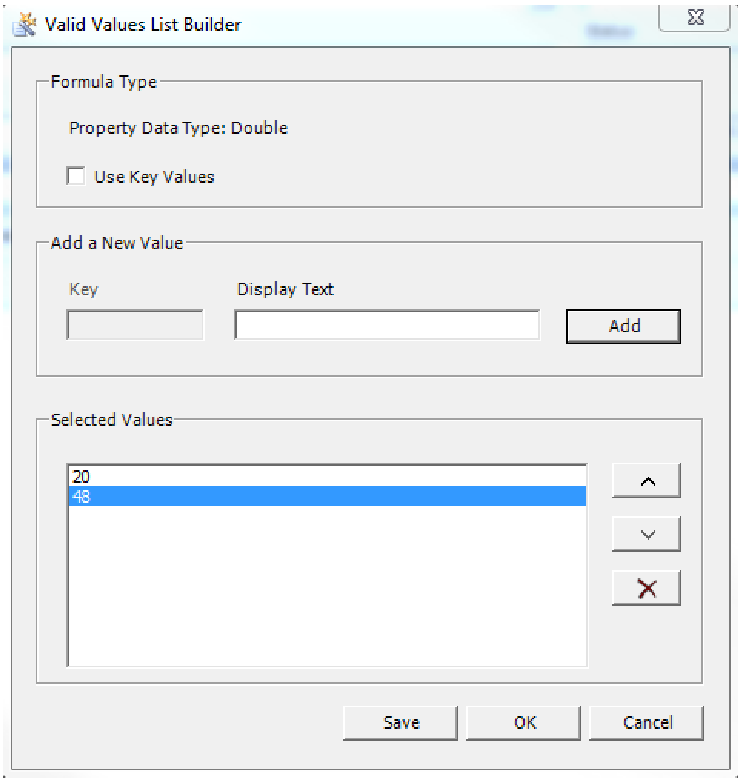
Constraint: ExamApp.Block

Formula | User Interface | Valid Values | When Changed | Display Value | Normalized Value | Delta Scoring | Database Constraint | Documentation

[✓ Standard] Auto Revert Follow Part Number Initialize Only Non-Cached

```
Result = Me.Length * Me.Width * Me.Height
```

Figure 9: Volume property with a formula for calculating the block volume



Valid Values List Builder

Formula Type

Property Data Type: Double

☐ Use Key Values

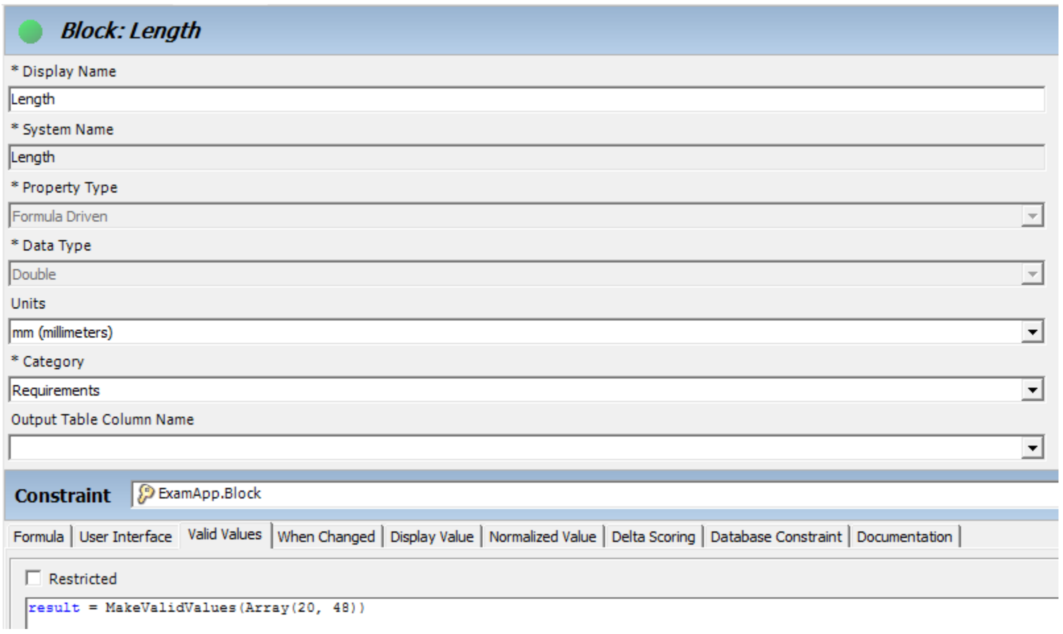
Add a New Value

Key	Display Text

Selected Values

20
48

Figure 10: Valid Values List Builder



Block: Length

* Display Name	Length
* System Name	Length
* Property Type	Formula Driven
* Data Type	Double
Units	mm (millimeters)
* Category	Requirements
Output Table Column Name	

Constraint ExamApp.Block

☐ Restricted

```
result = MakeValidValues(Array(20, 48))
```

Figure 11: Formula created with Valid Values List Builder

2.4.5 Processes

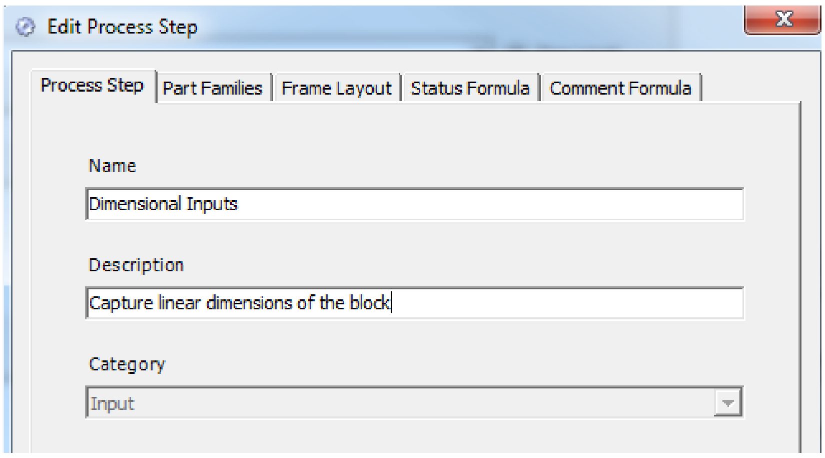
In the block example the user is prompted to specify the length, height and width, which is one process step. The volume property uses these properties to calculate the volume of the block in another process step. The visible and enabled checkbox option explain in user interface is controlling which properties are available in the process steps. Processes can only be promoted to a top-level part faimly. (Siemens PLM, 2017b)

A process step named dimensional inputs is made for the block part family. In the process step tab the users is able to specify a name and description for the process step, see Figure 12.

The part families tab allows the user to select the part families that contain the properties that should be displayed to the end user for this specific process step.

The user interface layout is selected in the frame layout tab. The user has multiple options but only two of them are illustrated here. The first option, **Auto Part Family Form**, is an automatically generated user interface, see Figure 13. The second option, **Custom Part Family Form**, allows placing of controls, such as buttons, labels, combo boxes, etc. on the user interface. (Siemens PLM, 2017b)

For the block example, Auto Part Family Form is selected, with the length, width, height and volume properties, see Figure 14.

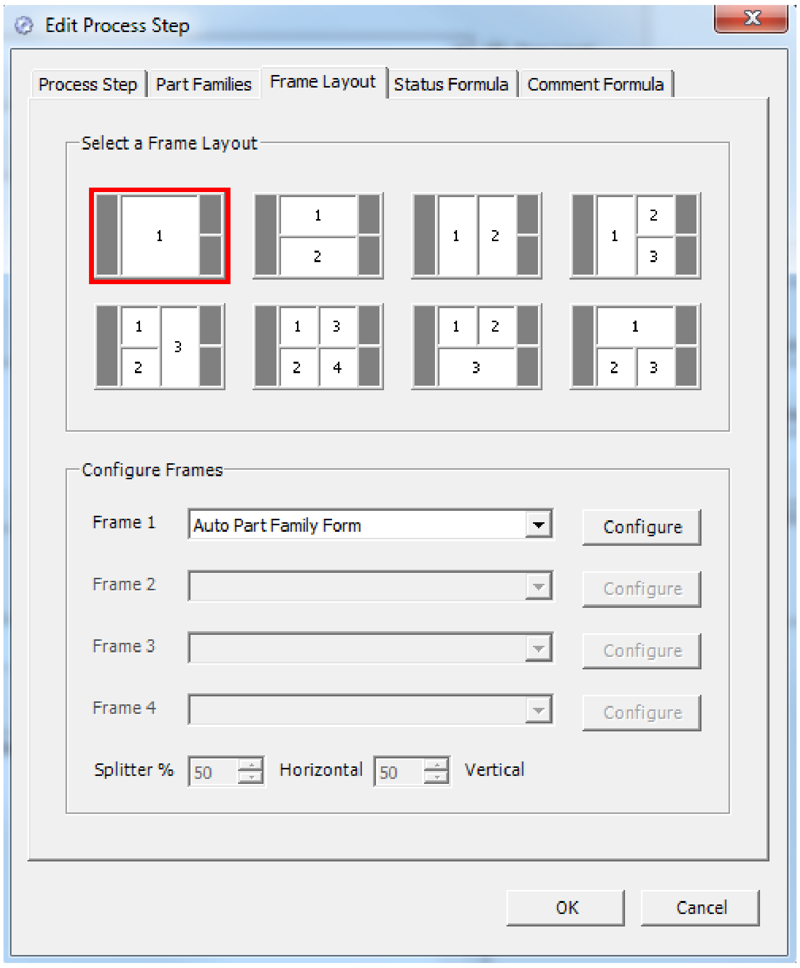


Name: Dimensional Inputs

Description: Capture linear dimensions of the block

Category: Input

Figure 12: Process step, dimensional inputs


Edit Process Step

Process Step | Part Families | Frame Layout | Status Formula | Comment Formula

Select a Frame Layout

Configure Frames

Frame 1: Auto Part Family Form — Configure

Frame 2: — Configure

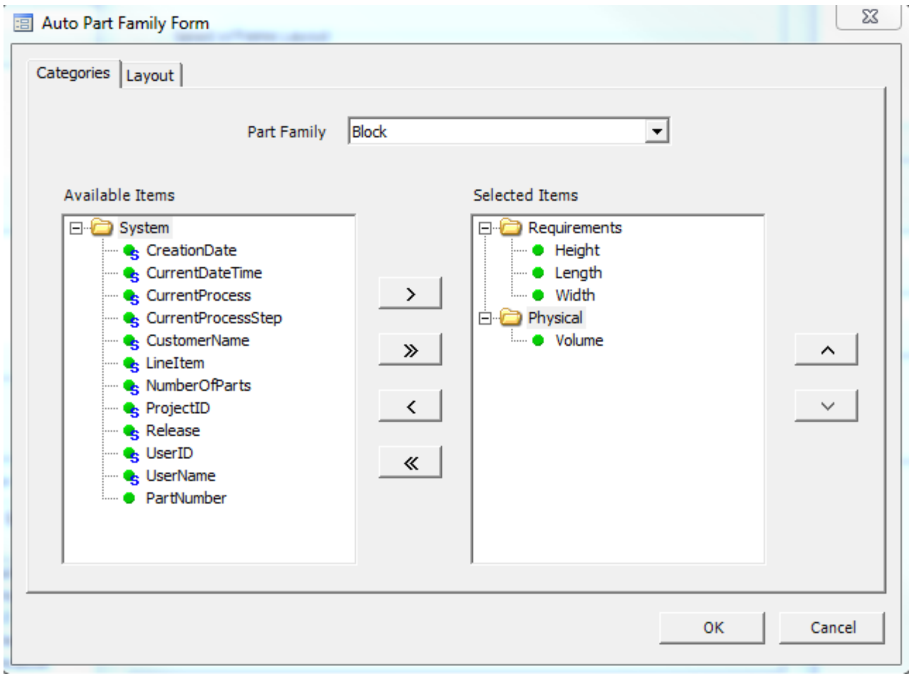
Frame 3: — Configure

Frame 4: — Configure

Splitter %: 50 Horizontal 50 Vertical

OK | Cancel

Figure 13: Frame layout


Auto Part Family Form

Categories | Layout

Part Family: Block

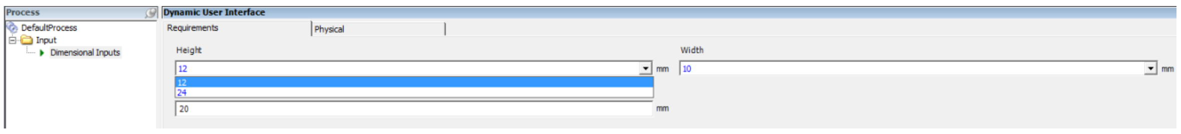
Available Items
- System
  - CreationDate
  - CurrentDateTime
  - CurrentProcess
  - CurrentProcessStep
  - CustomerName
  - LineItem
  - NumberOfParts
  - ProjectID
  - Release
  - UserID
  - UserName
  - PartNumber

Selected Items
- Requirements
  - Height
  - Length
  - Width
- Physical
  - Volume

OK | Cancel

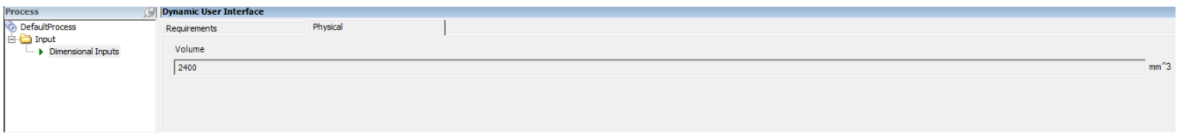
Figure 14: Selected properties

Now when the process step finished, the example application is ready to run in the Rulestream Engineer. When running the Rulestream Engineer, the user inputs a project name, e.g. example project, and selects which application, part family and process to use. After this, a new line item is created. The example application in Engineer run-time in Figure 15 and Figure 16.



Height: 12 mm (options: 12, 24)
Width: 10 mm
20 mm

Figure 15: Example application in the Rulestream Engineer



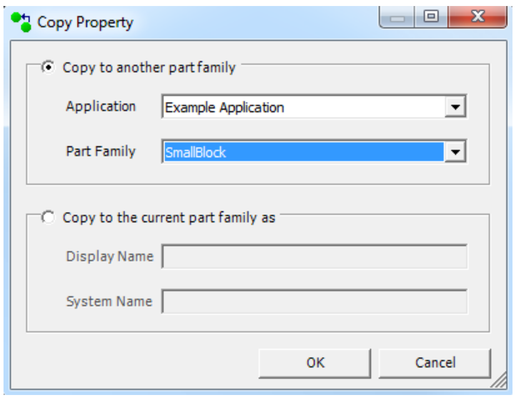
Volume: 2400 mm^3

Figure 16: Volume calculated from the dimensional inputs

2.4.6 Copying properties

Properties can be copied between part families in the same application or to another application in order to make the work more efficient. A property, for example the height property in the block part family can be copied to another part family that also needs a height property. (Siemens PLM, 2017b)

This is illustrated with a new small block part family in Figure 17 and Figure 18.



[✓ Copy to another part family]
Application: Example Application
Part Family: SmallBlock

Copy to the current part family as
Display Name:
System Name:

Figure 17: Creating small block part family

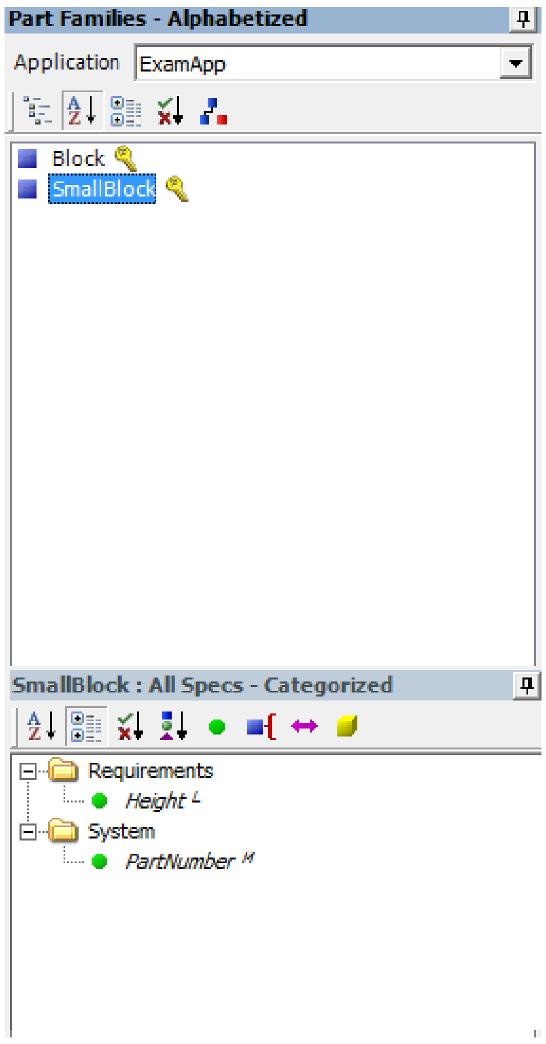
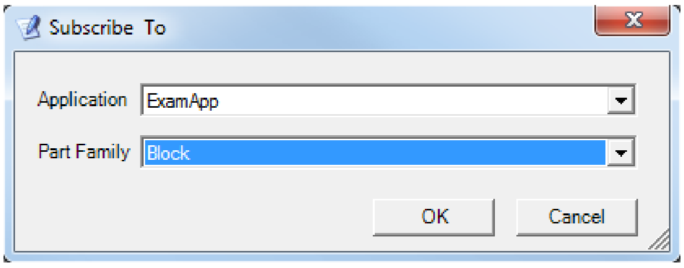


Figure 18: Height property copied to small block part family

2.4.7 Subscriptions

Part families can be subscribed (linked) to other part families. This means if the source property changes, then the subscribing property also changes. The user is also allowed to select/deselect which properties to subscribe to in the source part family. Subscribed properties can be deselcted after the subscribing process as well. This comes in handy when having multiple part families that uses the same property. (Siemens PLM, 2017b)

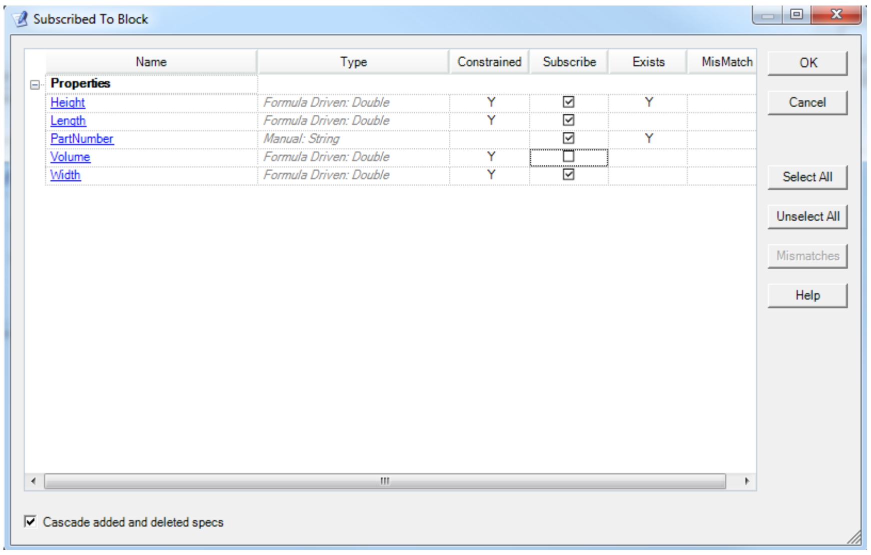
In the example, the small block part family is subscribed to the block part family, with volume property deselected. See Figure 19, Figure 20 and Figure 21 for the example.



Application: ExamApp

Part Family: Block

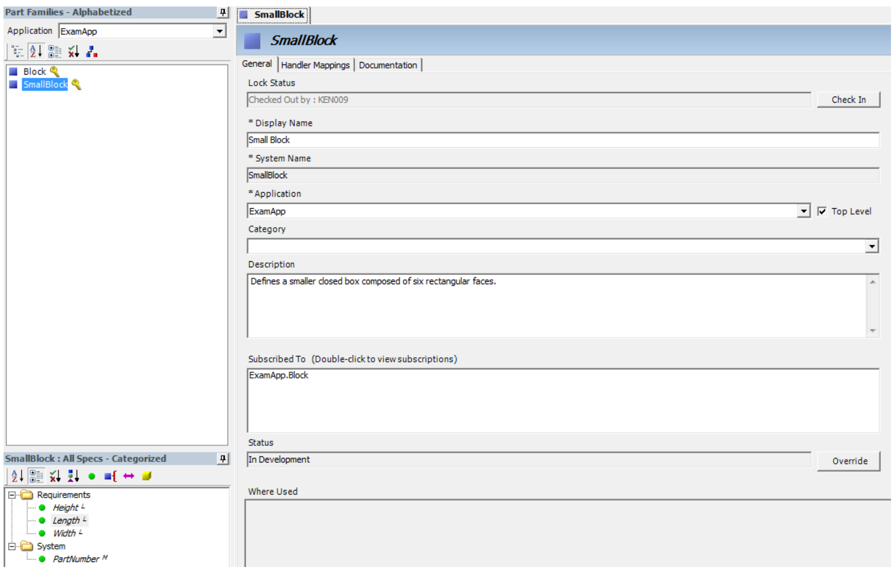
Figure 19: Subscribing small block part family to block part family



Name	Type	Constrained	Subscribe	Exists	MisMatch
Properties					
Height	Formula Driven: Double	Y	☑	Y	
Length	Formula Driven: Double	Y	☑		
PartNumber	Manual: String		☑	Y	
Volume	Formula Driven: Double	Y	☐		
Width	Formula Driven: Double	Y	☑		

☑ Cascade added and deleted specs

Figure 20: Selection/deselection of subscription properties



Application: ExamApp

Lock Status: Checked Out by : KEN009

* Display Name: Small Block

* System Name: SmallBlock

* Application: ExamApp ☑ Top Level

Category:

Description: Defines a smaller closed box composed of six rectangular faces.

Subscribed To (Double-click to view subscriptions): ExamApp.Block

Status: In Development

Where Used:

Figure 21: Small block part family subscribed to block part family

2.4.8 Subpart collections

A part family can have multiple part families related to itself. The subpart collections is a parent/child relataionship between these to identify which part family is the parent and which is the child. This promotes a tree structure of part families. (Siemens PLM, 2017b)

To illustrate this an example of a table part family is used. The table has a table top, legs and wheels, for which part families are created as well. Subpart collections named **surface** and **supports** are created. Child part families are selected. Another subpart colletion named **wheel** is created for the child part family **leg**. To this subpart collection, the child part family **caster** is selected. This is seen in Figure 22.

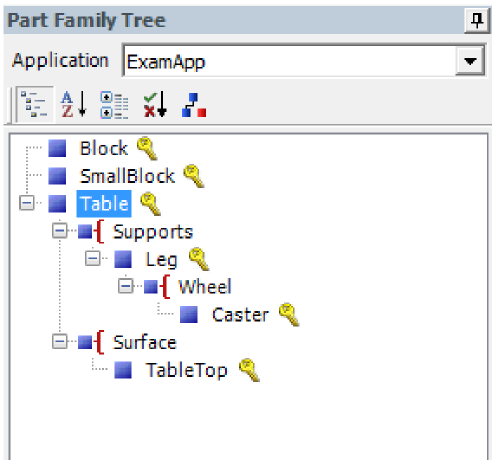
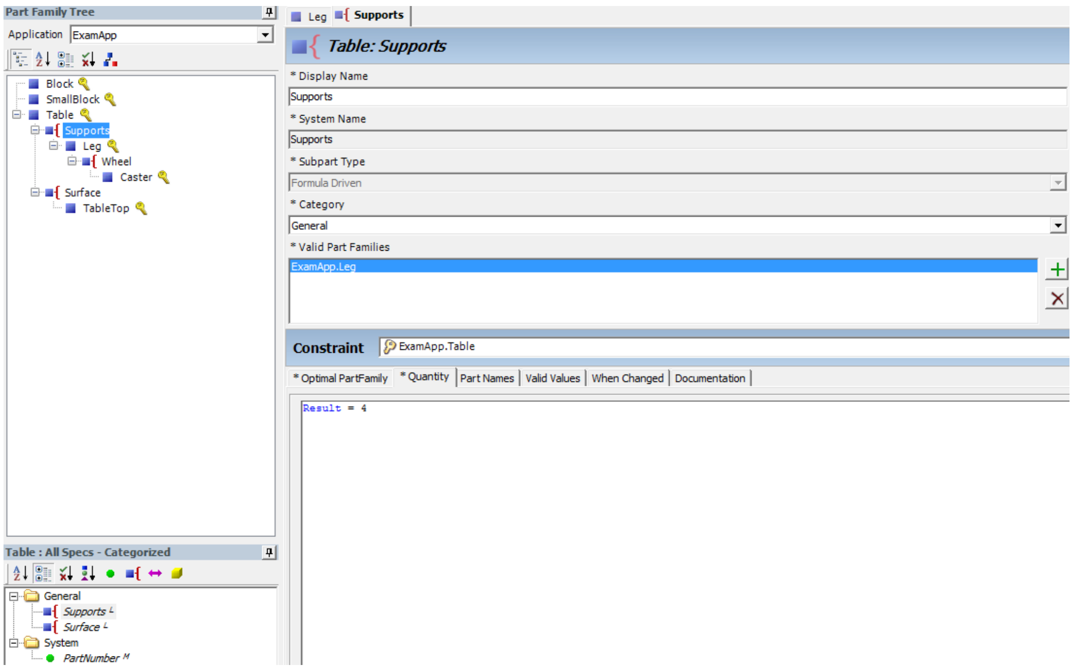


Figure 22: Table part family with subpart collections

In the subpart collection **supports**, there is a tab named **quantity**. The user has to possibility to create a formula for the quantity of the subparts. In this example, the quantity is set to “4”, as can be seen in Figure 23.



* Display Name: Supports

* System Name: Supports

* Subpart Type: Formula Driven

* Category: General

* Valid Part Families: ExamApp.Leg

Constraint: ExamApp.Table

```
Result = 4
```

Figure 23: Quantity tab in supports subpart collection

A process is created with the Auto Part Family Form that was explained earlier. In Figure 24 is the model in run-time.

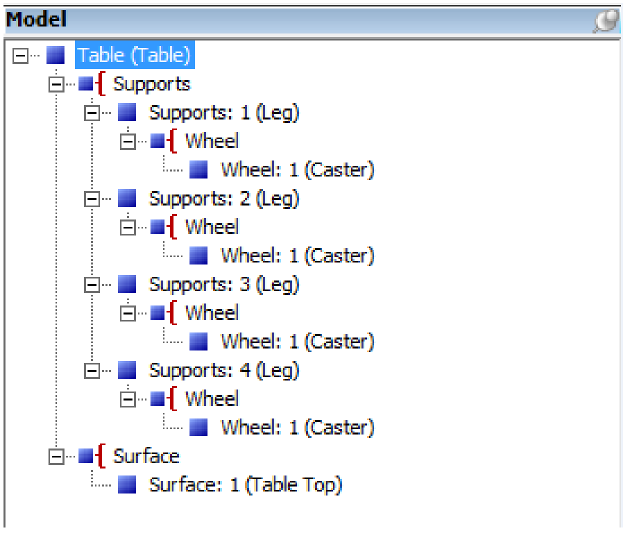


Figure 24: Table part family in run-time

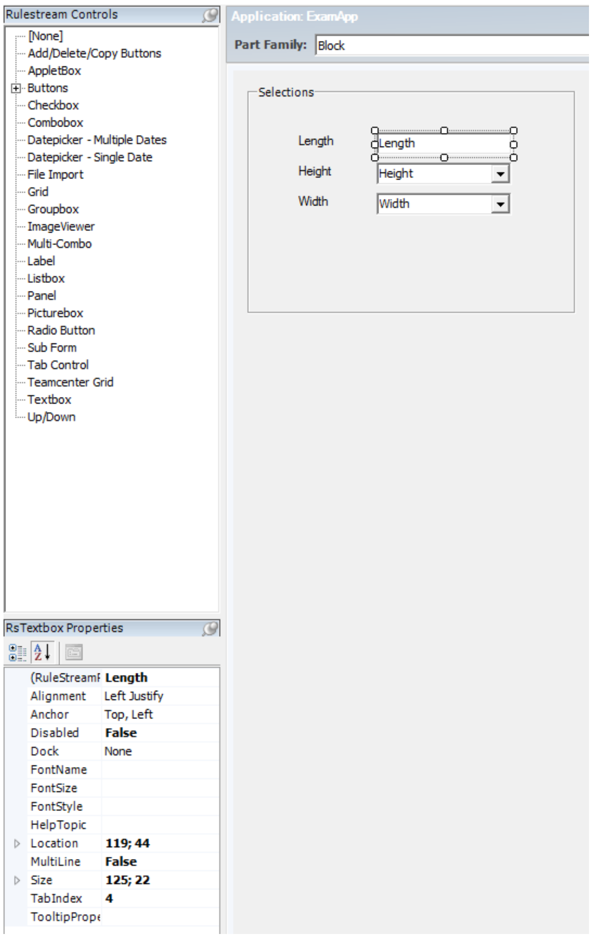
2.4.9 The custom UI designer

The custom user interface designer consists of two selections, **Custom Part Family Form** and **Custom Control**.

The **Custom Part Family Form** allows placing of controls, such as buttons, labels, combo boxes, etc. on the user interface.

The **Custom Control** is a Visual Basic.NET control, which will not be further explained in this thesis. (Siemens PLM, 2017b)

An example of a custom user interface. The interface is made for the block part family with Custom Part Family Form. The process for the block is reused from previous examples. In the custom user interface, labels for the length, height and width are used. For the length specification, a text box is used. For the height and width, comboboxes are used. Rulestream properties are selected for the textbox and combo boxes. The creation of the custom interface is seen in Figure 25. The custom interface in run-time is shown in Figure 26.


Rulestream Controls

- [None]
- Add/Delete/Copy Buttons
- AppletBox
- Buttons
- Checkbox
- Combobox
- Datepicker - Multiple Dates
- Datepicker - Single Date
- File Import
- Grid
- Groupbox
- ImageViewer
- Multi-Combo
- Label
- Listbox
- Panel
- Picturebox
- Radio Button
- Sub Form
- Tab Control
- Teamcenter Grid
- Textbox
- Up/Down

Application: ExamApp

Part Family: Block

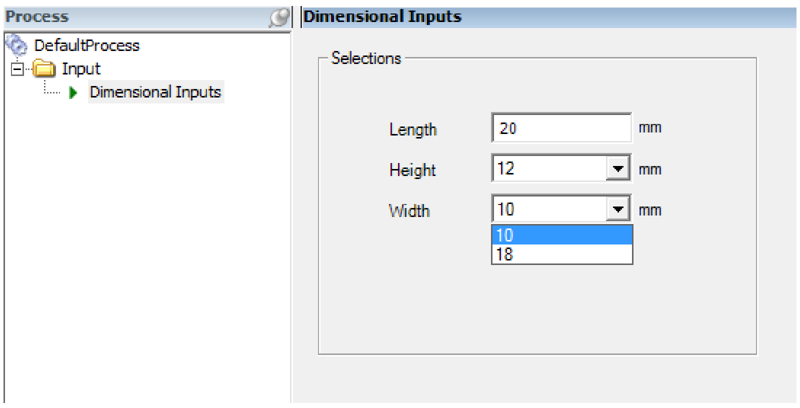
Selections

Length	Length
Height	Height
Width	Width

RsTextbox Properties

(RuleStreamF	**Length**
Alignment	Left Justify
Anchor	Top, Left
Disabled	**False**
Dock	None
FontName	
FontSize	
FontStyle	
HelpTopic	
Location	**119; 44**
MultiLine	**False**
Size	**125; 22**
TabIndex	**4**
TooltipPrope	

Figure 25: Custom user interface for the block part family



Length 20 mm
Height 12 mm
Width 10 mm
10
18

Figure 26: The custom user interface in run-time

2.4.10 Connections

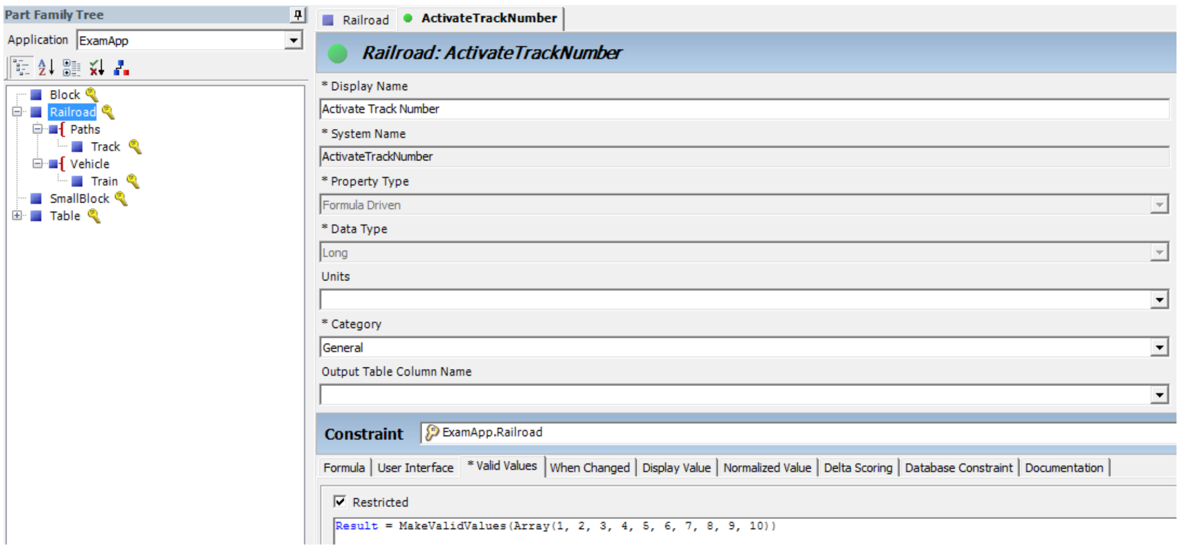
A connection is similar to a subpart collection. The difference with the connection is that the parent part family does not own the subpart. The connection defines a "knows-a" relationship between part families. Because of that relationship, any part family can have a relationship with any other part family in the product control model. (Siemens PLM, 2017b)

A train's engine size may depend on the maximum slope of a track. Each track of a railroad may have a different slope property. For the train to know the value of the slope property depending on which track the train is currently on, connections can be used.

To show how to use connections, an example with a railroad consisting of ten tracks and a train is given.

The railroad (parent part family) is the top-level part family. It has two subpart collections, paths and vehicle. To these subpart collections, valid part families are selected (child part families). For the paths subpart collection, the selection made is "Tracks" and for the vehicle, subpart collection the selection made is "Train".

A property called Activate Track Number is added to the Railroad part family. For this property, a formula is written, which creates a valid values selection from 1-10, see Figure 27. This property is used in run-time to select on which track the train rolls.



Application: ExamApp

* Display Name: Activate Track Number

* System Name: ActivateTrackNumber

* Property Type: Formula Driven

* Data Type: Long

Units:

* Category: General

Output Table Column Name:

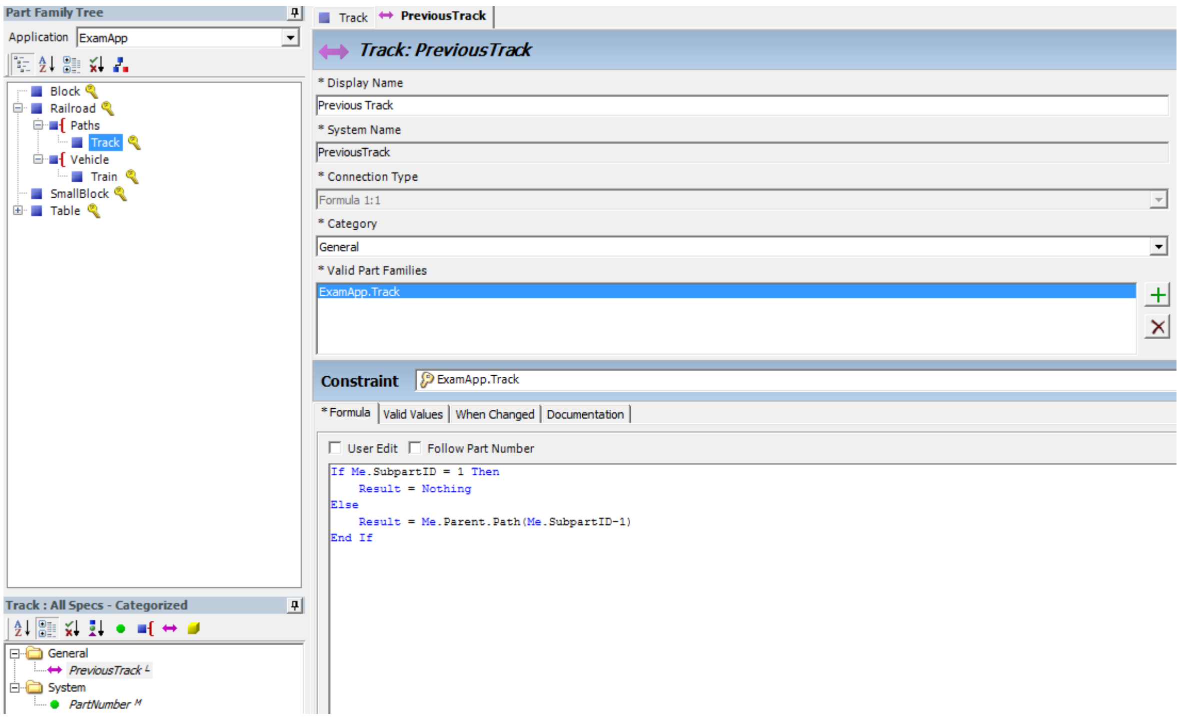
Constraint: ExamApp.Railroad

☑ Restricted

```
Result = MakeValidValues(Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10))
```

Figure 27: Valid values formula

The quantity of tracks is set to 10 in the quantity tab in the paths subpart collection. The tracks connects to each other with a connection named previous track in the track part family, see Figure 28.



Application: ExamApp

* Display Name: Previous Track

* System Name: PreviousTrack

* Connection Type: Formula 1:1

* Category: General

* Valid Part Families: ExamApp.Track

Constraint: ExamApp.Track

☐ User Edit ☐ Follow Part Number

```
If Me.SubpartID = 1 Then
    Result = Nothing
Else
    Result = Me.Parent.Path(Me.SubpartID-1)
End If
```

Figure 28: Previous track connection

The user can select between different types of connections in the connection type field, see Figure 28. The selections are either **manual** or **formula**.

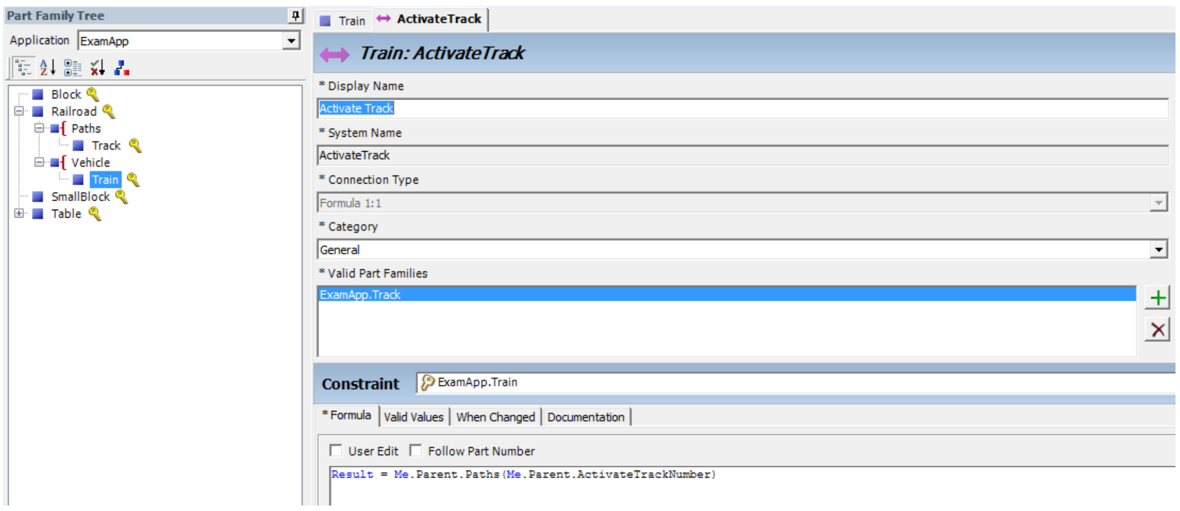
Manual means that the part or parts can be connected manually through the user interface.

Formula means that the part or parts can be connected and disconnected via formula or user interface. (Siemens PLM, 2017b)

Connection types:

- Manual 1:1 – One-to-One connection
- Manual 1:M – One-to-Many connection
- Formula 1:1 – One-to-One connection
- Formula 1:M – One-to-Many connection

One last connection named Activate Track is added to the train part family, with the track as the valid part family. In the formula, a connection to the track is made that corresponds to the value of the Activate Track Number property in the parent part family, see Figure 29.



* Display Name: Activate Track

* System Name: ActivateTrack

* Connection Type: Formula 1:1

* Category: General

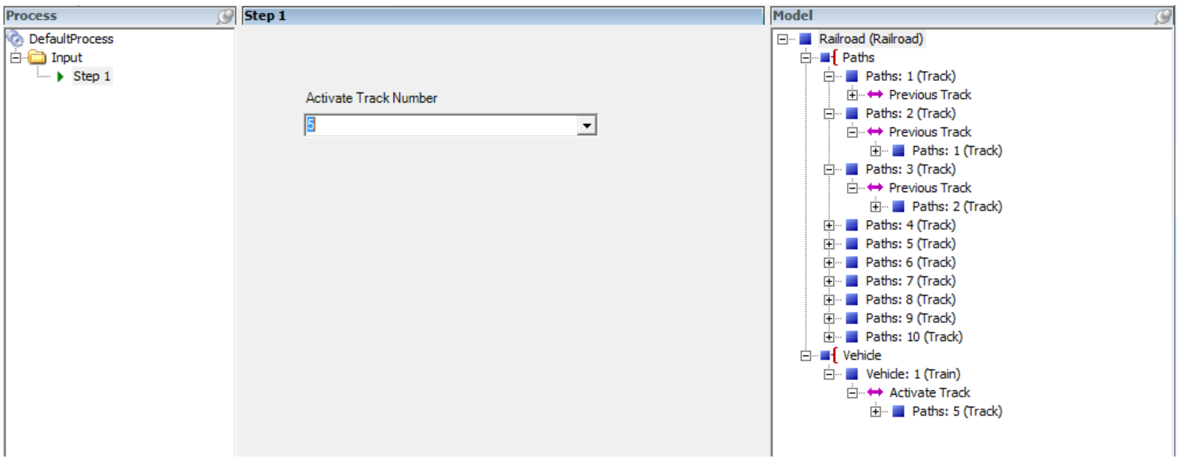
* Valid Part Families: ExamApp.Track

User Edit [ ] Follow Part Number [ ]

```
Result = Me.Parent.Paths(Me.Parent.ActivateTrackNumber)
```

Figure 29: Activate track connection formula

A process is created for the railroad part family, and for the user interface, the Auto Part Family Form is selected. Figure 30 shows the model in run-time.



```
Process
DefaultProcess
  Input
    Step 1

Step 1
Activate Track Number
5

Model
Railroad (Railroad)
  Paths
    Paths: 1 (Track)
      Previous Track
    Paths: 2 (Track)
      Previous Track
        Paths: 1 (Track)
    Paths: 3 (Track)
      Previous Track
        Paths: 2 (Track)
    Paths: 4 (Track)
    Paths: 5 (Track)
    Paths: 6 (Track)
    Paths: 7 (Track)
    Paths: 8 (Track)
    Paths: 9 (Track)
    Paths: 10 (Track)
  Vehicle
    Vehicle: 1 (Train)
      Activate Track
        Paths: 5 (Track)
```

Figure 30: Railroad model in run-time

Connections promote the modularity concept used to build configurable applications. They form references between part families. Connections are commonly used to relate “previous” or “next” instances. Benefits of connections are reduction of context overrides and avoidance of long reference chains. (Siemens PLM, 2017b)

2.4.11 Siemens NX assemblies

Each CAD part in Rulestream is represented by a NX specification. During run-time, Rulestream will use the specification information to build the CAD assembly in NX. The part that is scanned in is not necessarily the final part; the part ID can be substituted during by specifying the part ID at run-time. This is the method used in the W31 Application.

3 Methodology

In this chapter, I explain the work process, from starting the project to creating the 3D configurator logics. This includes meetings, interviews and methods used.

3.1 Project start

The project started with research about the current state of usage of 3D configurators. My supervisor and I arranged a meeting with project manager for the W32 design team, Harald Slotte. We interviewed him and we were able to find out that several 3D configurators were either in use, or under development.

One existing 3D configurator is the Parametric configurator, which can generate simplified engine models based on another configurator called IOS (Internal Order Specification) configurator. The parametric configurator automatically creates simplified engine models of W20, W32 and W34 engines. In the IOS configurator, the user can specify different configurations or a project specific configuration, which in turn is loaded into the parametric configurator.

CGS is another 3D configurator. It uses models created in the absolute zero point. This means that when creating the model in the Siemens NX, the designer has to make sure that it modelled in the correct location according to the engine. By having all the models modelled in the absolute zero point, using suppression rules in Siemens NX is possible to show or hide the correct module variants. The development of this configurator is however on hold and does not support modular architecture very well.

My supervisor and I decided that we had to do more research. He gave me a list with names of persons that might have knowledge and information in the configurator area.

The first person that I contacted was Juhani Sahlberg. He provided me with some information about how the configurators receives information. The configurators uses data in the form of Engineering Bill of Materials (EBoM).

Harri Piili introduced me to the creation of EBoM’s, and the processing of them within Teamcenter. Teamcenter is a PLM (Product Lifecycle Management) software used to store drawings and models. After getting a little more knowledge about this, he informed me about a project called Rulestream. The contact person for this project was Petri Pättiniemi, Modular Platform expert.

3.2 Rulestream project introduction

The previous summer when I was working at Wärtsilä, I was creating simplified models of the W31 module variants. The simplified models had to be made in a specific way so that they later could be used in a 3D configurator that was going to be Rulestream at that point. Due to this, I was already somewhat familiar with the Rulestream project.

In a meeting with Petri Pättiniemi, I presented the goal of the Bachelor´s thesis for him. He then informed about the current state of the project, and that recently they decided that they are purchasing licenses for the Rulestream software. This was the perfect opportunity to make my thesis a part of the project. We scheduled another meeting with Franco, Petri and

I, to define the scope of the thesis, and agree that I would do my thesis as a part of the project. The scope was then limited to doing a user survey, a data shape assessment of the modules for the W31 and creating the logics for the 3D configurator.

3.3 User survey

The use cases of a 3D configurator are many. In order to get a better view of the use cases, I did a user survey with the stakeholders. My supervisor provided me with names of persons from different areas in Wärtsilä, which he believed would be interested in this kind of tool.

3.3.1 Potential users

- Project engine room systems
- Sales material
- Delivery Center Vasa (DCV)
- Design W31
- Wärtsilä Land and Sea Academy (WLSA)

I contacted the persons from the different areas and scheduled meetings. The meetings were either Skype meetings or meetings in person. To be able to present the scope of the project, I prepared a PowerPoint presentation.

3.4 Data shape assessment

In order to make the logics work for selecting and positioning the W31 module variants, each component has to have interfaces in the model structure. These interfaces are connecting points, acting as “male” and “female” connectors. The relations between these connecting points are specified in module report documents, which are maintained when variants are revised or new designs are released.

The data shape of the module variants has to be checked manually. This is done by opening the module variants in NX. The user can specify which parts, sketches, coordinate systems, etc. should be visible in NX, by creating different reference sets. The user can then select between the reference sets, and the part is then visualized the way the reference set is made. The module variants should contain following reference sets:

- MODEL – includes all components in the module variant.
- MODULE VARIANT – includes all components and connecting points.
- MODULE SKELETON – includes only the connecting points.
- REPRESENTATION – includes representations (lightweight components) and connecting points.

The connecting points in the module variants needs to be tested to verify that they are correctly positioned. This also has to be done manually in NX. The behavior of the connection points are tested by adding them to an assembly with NX assembly constraints. For every module, there are documents called Module reports, with information about the module as well as how the connecting points are related to connecting points in other modules.

The results of the data shape assessment had to be stored, so that module variants in bad shape can be fixed later. I decided to prepare and Excel spreadsheet, with the module variants listed. This way, I could make notes about which connecting points I chose to use, as well as explain errors and problem that I found, in more detail.

3.5 Rulestream workshop

Before starting to use the Rulestream software, Lauri Majamaa from IDEAL hosted a training session at Wärtsilä in Vaasa. In this training session, Solution Manager Petri Bergius from Wärtsilä attended as well, to support us with Teamcenter and Siemens NX related matters.

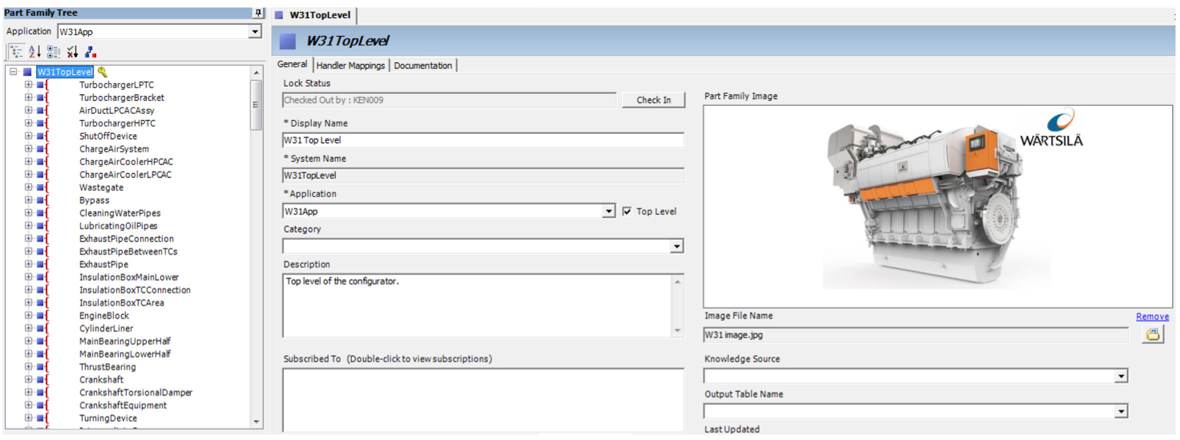
The Rulestream training is usually one week long and more in depth. Due to limited time and other reasons, this training session only lasted one day. The scope of the training was to get used to the features within Rulestream. Lauri went through some examples of how to use the features, by taking some basic engine components and create the logics for them. We created an NX specification and a simple user interface to use for testing that the positioning logics of the module variants in an assembly in Siemens NX works.

3.6 The W31 Application

This chapter describes the different features used and creation of the logics in Rulestream for the 3D configurator.

3.6.1 The W31 Application structure

To start working in the Rulestream environment, an application is needed. I decided to name it “W31App”. In the W31App, I created a top-level part family that I named “W31TopLevel”. Below the W31TopLevel I added subpart collections and named them corresponding to the module names. This formed a tree structure, see Figure 31.


Part Family Tree

Application: W31App

- W31TopLevel
  - TurbochargerLPTC
  - TurbochargerBracket
  - AirDuctLPCACAssy
  - TurbochargerHPTC
  - ShutOffDevice
  - ChargeAirSystem
  - ChargeAirCoolerHPCAC
  - ChargeAirCoolerLPCAC
  - Wastegate
  - Bypass
  - CleaningWaterPipes
  - LubricatingOilPipes
  - ExhaustPipeConnection
  - ExhaustPipeBetweenTCs
  - ExhaustPipe
  - InsulationBoxMainLower
  - InsulationBoxTCConnection
  - InsulationBoxTCArea
  - EngineBlock
  - CylinderLiner
  - MainBearingUpperHalf
  - MainBearingLowerHalf
  - ThrustBearing
  - Crankshaft
  - CrankshaftTorsionalDamper
  - CrankshaftEquipment
  - TurningDevice

W31TopLevel

W31TopLevel

General | Handler Mappings | Documentation

Lock Status: Checked Out by : KEN009

* Display Name: W31 Top Level

* System Name: W31TopLevel

* Application: W31App [✓ Top Level]

Category:

Description: Top level of the configurator.

Subscribed To (Double-click to view subscriptions)

Part Family Image

Image File Name: W31 image.jpg

Knowledge Source:

Output Table Name:

Last Updated

Figure 31: The W31App application

The subpart collections need to have at least the following formula driven properties defined:

- ItemID – Property that contains Visual Basic code that defines the item ID of the module variants.
- Constraints – Property that holds the logics for the module connecting points.

Rulestream has the ability to add NX components to an existing empty assembly. In order for Rulestream to add the components into the assembly, the top-level part family must have an NX specification, for explanation see chapter 2.4.11. This NX specification is added to the W31TopLevel part family and has the assembly template.

The subpart collections below the W31TopLevel part family also need to have NX specifications. The NX specification added to the subpart collections are different from the one added to the W31TopLevel part family.

The most efficient way to get these properties and the NX specification into every subpart collection was to create another application in Rulestream. I named this application “CMNApp” (Common App). In this application, I created a top-level part family that I named GenericNXComponent, where I also created the properties “ItemID” and “Constraints”, and the NX specification, see Figure 32. At the same time that the subpart collections were created, they were also subscribed to the CMNApp. This way they received all the listed properties and the NX specification.

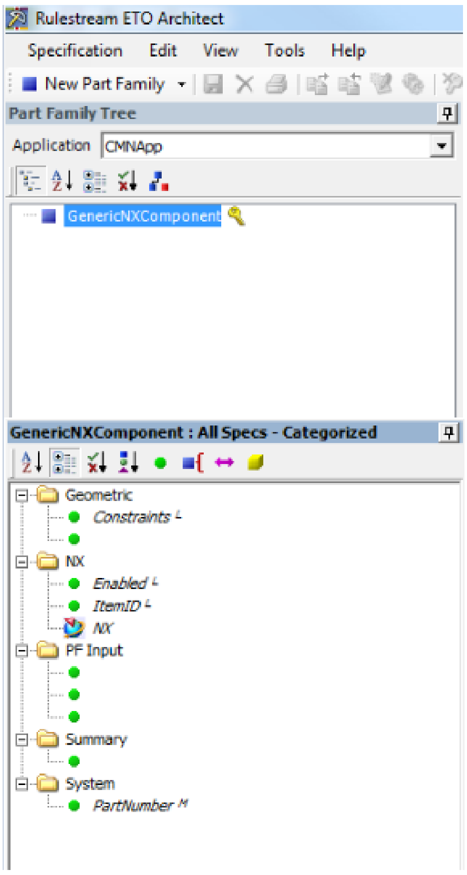


Figure 32: The CMNApp application

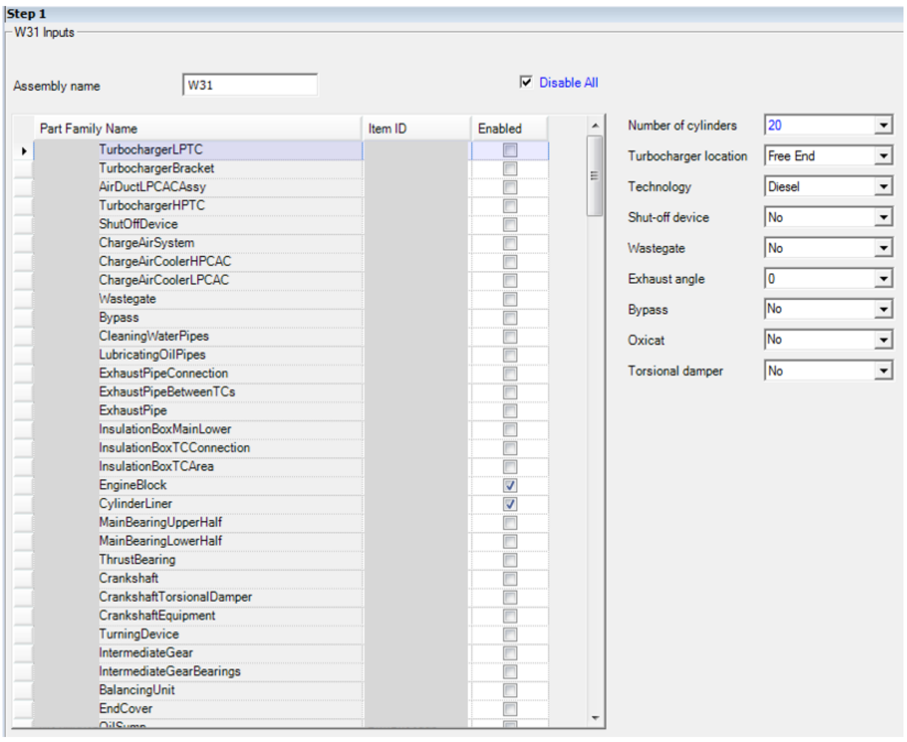
3.6.2 The user interface

The logics need to be tested. To be able to do this, there is a need of a user interface that allows me to configure the engine in different ways. Selection and positioning of the engine components depends on the engine configuration. A few examples of different selections for an engine configuration are:

- Number of cylinders: 8, 10, 12, 14, 16 or 20
- Technology: Diesel, Dual Fuel or Gas

- Turbocharger location: Free end or Driving end
- Oil sump: Wet sump, Dry sump or Deep wet sump
- Single main engine: Yes or No

The user interface is created with the Rulestream feature called “The custom UI Designer”. For explanation of this feature, see chapter 2.4.9. A part of the user interface is shown in Figure 33. The module numbers and the item ID´s are not shown because it is classified information.


Step 1

W31 Inputs

Assembly name: W31

☑ Disable All

Part Family Name	Item ID	Enabled
TurbochargerLPTC		☐
TurbochargerBracket		☐
AirDuctLPCACAssy		☐
TurbochargerHPTC		☐
ShutOffDevice		☐
ChargeAirSystem		☐
ChargeAirCoolerHPCAC		☐
ChargeAirCoolerLPCAC		☐
Wastegate		☐
Bypass		☐
CleaningWaterPipes		☐
LubricatingOilPipes		☐
ExhaustPipeConnection		☐
ExhaustPipeBetweenTCs		☐
ExhaustPipe		☐
InsulationBoxMainLower		☐
InsulationBoxTCConnection		☐
InsulationBoxTCArea		☐
EngineBlock		✓
CylinderLiner		✓
MainBearingUpperHalf		☐
MainBearingLowerHalf		☐
ThrustBearing		☐
Crankshaft		☐
CrankshaftTorsionalDamper		☐
CrankshaftEquipment		☐
TurningDevice		☐
IntermediateGear		☐
IntermediateGearBearings		☐
BalancingUnit		☐
EndCover		☐

Number of cylinders	20
Turbocharger location	Free End
Technology	Diesel
Shut-off device	No
Wastegate	No
Exhaust angle	0
Bypass	No
Oxicat	No
Torsional damper	No

Figure 33: The W31App user interface in Rulestream Engineer

3.6.3 Creating the logics

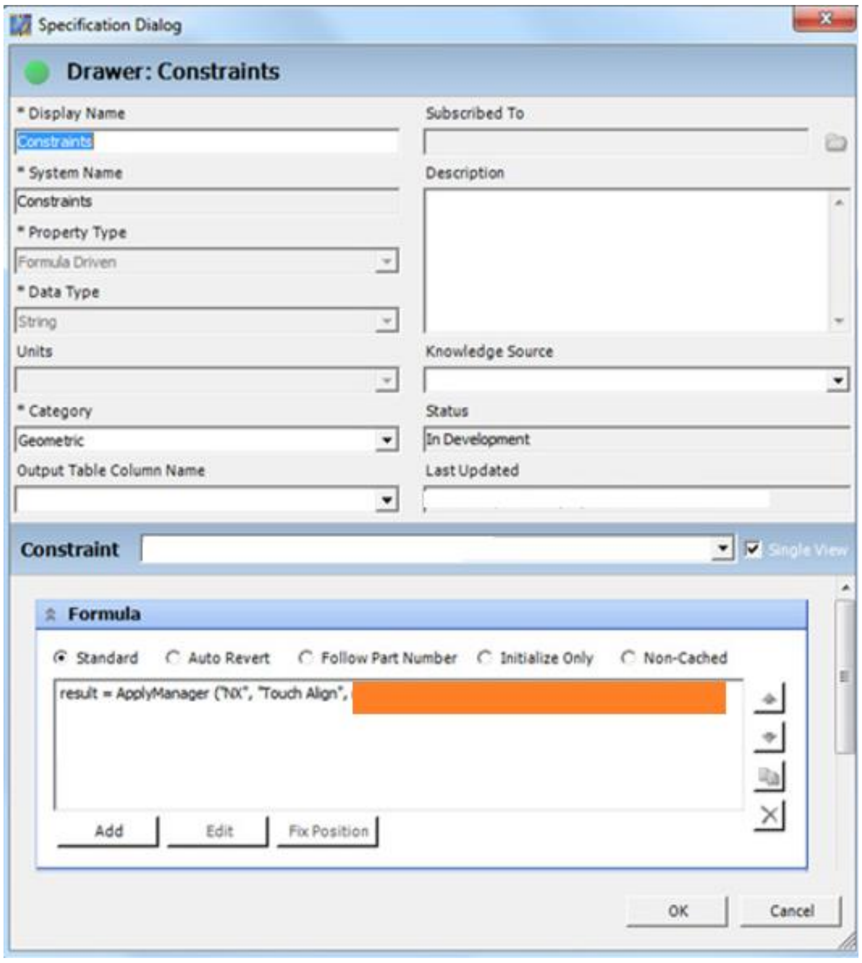
To start creating the logics, I had to decide which component was the best one to start with. This was not difficult, as the engine block is used almost in every engine assembly and it has the largest amount of connecting points compared to all the other modules.

One requirement when adding a component to the assembly is to locate the component in the NX model space. Rulestream uses the following NX constraining methods:

- Touch & Align– Locates two objects coincident with each other by locating them so that their normals point in opposite directions.

- Parallel – Constraint that enables to position two objects to lie parallel to each other.
- Perpendicular – Constraint that enables to make to objects to lie perpendicular to each other.
- Distance – Constraint that allow offsetting the object being mated from the object that is being mated to them with a distance.
- Angle – Constraint to allow positioning two objects by defining a rotational angle between them.

The Constraint property of every subpart collection is where the mating logic is defined. The property is a string type, but the formula cannot be modified or edited by typing in the formula box. The mate condition used by NX must be constructed with the Mate builder. In Figure 34, a new mate condition is added by clicking the "Add" button. The mate condition is already added in the CMNApp, to which the subpart collections are subscribed. By clicking the edit button, I modified the condition for each subpart collection.


Specification Dialog

Drawer: Constraints

* Display Name: Constraints	Subscribed To:
* System Name: Constraints	Description:
* Property Type: Formula Driven	
* Data Type: String	
Units:	Knowledge Source:
* Category: Geometric	Status: In Development
Output Table Column Name:	Last Updated:

Constraint [✓ Single View]

Formula

[✓ Standard] Auto Revert Follow Part Number Initialize Only Non-Cached

```
result = ApplyManager ("NX", "Touch Align", [redacted]
```

Figure 34: Constraint property for a subpart collection

The engine block is always added and locked to the absolute origin in the NX assembly with a fixed constraint. The other modules are added to the NX assembly with the “Touch & Align” constraint. Mating is accomplished with the NX mate builder. The form includes four different sections, seen in Figure 35.

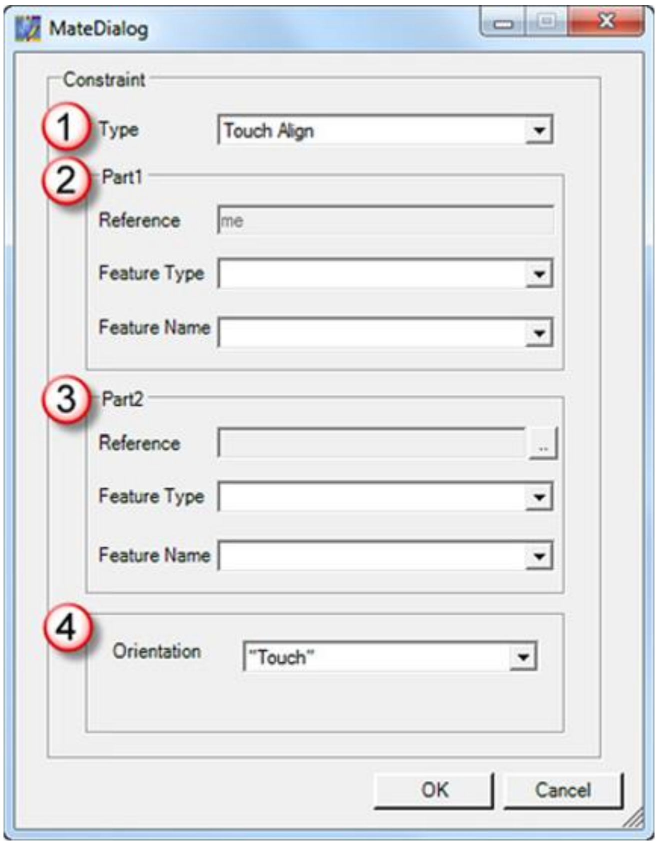


Figure 35: NX mate builder

1. Type – The type of constraint to be created.
2. Part1 – The current subpart collection.
3. Part2 – The mating subpart collection.
4. Orientation – Specifies a “Touch” or “Align” mate. Faces and edges cannot be used to mate.

The Feature type (type of connecting point) and Feature name (name of the connecting point) for Part 1 and Part 2 is defined. It is not revealed which feature type that is used in this thesis, as it is classified information. The available feature types are:

- Plane – Datum planes can be rotated in any direction. Since faces cannot be used with the Mate builder, datum planes can be specified on faces in NX components.
- Axis – An axis can point in any direction. To align two NX components, axis's can be used.
- Point – Points do not have any direction. If the mating NX components are to be aligned, a minimum of two points have to be used.
- Csys – Coordinate systems in NX components can be used to orient and position NX components.

Now that the logics for mating and positioning are created, I need to create the logics for module variant selection and quantity, to be able to test the mating and positioning logics. For example, if I want configure the engine to be diesel- or gas-driven, and 8 or 10 cylinders, the cylinder head module variant will change, as well as the quantity of the cylinder heads.

```
Select Case Me.VariantType
        Case "V1"
                Result = "ABCD123456"
        Case "V2"
                Result = "ABCD234567"
End Select
```

Figure 36: Formula written in the ItemID property

The ItemID formula in Figure 36, stores the item ID of module variant 1 in case "V1", and the item ID of module variant 2 in case "V2".

```
Select Case Me.Technology
        Case "Diesel"
                Result = "V1"
        Case "Gas"
                Result = "V2"
End Select
```

Figure 37: Formula written in the VariantType property

The VariantType formula in Figure 37, selects the module variant “V1” if the engine is diesel-driven. If it is gas-driven, it selects module variant “V2”.

```
Select Case Me.NumberOfCylinders
        Case 8
                Result = 8
        Case 10
                Result = 10
        Case 12
                Result = 12
        Case 14
                Result = 14
        Case 16
                Result = 16
        Case 20
                Result = 20
End Select
```

Figure 38: Formula for the quantity of cylinder heads.

The formula in Figure 38, selects the quantity of the cylinder heads, depending on the selected number of cylinders. This formula is written in the Quantity field in the cylinder head subpart collection.

For these formulas to work, I had to create additional properties, that I named “NumberOfCylinders” and “Technology”, and made them enabled and visible in the user interface. The properties contains Visual Basic language for a valid values array formula. By making these properties enabled and visible, I am able to select between the valid values in the user interface.

The formulas in Figure 39 and Figure 41 selects the first value available from the formulas in Figure 40 and Figure 42.

```
Result = Me.ValidValues("NumberOfCylinders").FirstValue
```

Figure 39: Formula in the Formula tab for the NumberOfCylinders property

```
Result = MakeValidValues(Array(8, 10, 12, 14, 16, 20))
```

Figure 40: Formula in the Valid Values tab for the NumberOfCylinders property

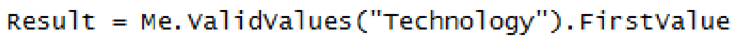

```
Result = Me.ValidValues("Technology").FirstValue
```

Figure 41: Formula in the Formula tab for the Technology property

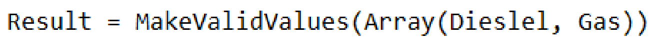

```
Result = MakeValidValues(Array(Dieslel, Gas))
```

Figure 42: Formula in the Valid Values tab for the Technology property

These formulas are used just as an example. More in depth description of the actual formulas are not shared in this thesis, as it is classified information.

4 Results

This chapter presents the results of the thesis work. This includes the results from the user survey, the data shape assessment and the creating the logics.

4.1 User survey

This sub-chapter presents the results from the interviews done in the user survey.

4.1.1 Project Engine Room Systems

A big portion of this department's work consists of making 2D drawings. Their customers use these for example when planning an engine room. The 2D drawings can be sales drawings, main dimension drawings for different engine configurations and drawings with the engine and a generator mounted on a common base frame (Genset drawings). These drawings do not have to be very detailed. The most important thing is that the main dimensions are correct. To make the work required less time consuming, they use simplified 3D engine models, as they require less computer capacity to handle.

The importance of fast generation of 2D drawings is very high, due to the amount of sales quotations each month. This process is kept fast by using the Parametric configurator to generate simplified engine models. To keep the configurator up to date with new designs, the team maintains it with new simplified models and logics for positioning. (Södö, 2017)

It would be beneficial to have both the simplified and fully detailed 3D engine in one configurator, to avoid maintenance of multiple ones.

4.1.2 Sales material

3D engine models simplifies the communication with customers by giving them a “hands on feeling”, which means they get a better view of what they are purchasing. Visually appealing sales material such as brochures and posters are highly valuable. This is achievable by having high detailed 3D engine models. However, building the 3D engine models is time consuming and requires a lot of work.

When doing the animation work for the sales material, the assembling of the 3D engine model is done manually. In many cases when the assembly of the model is done, there might have been new designs released, and the model is no longer up-to-date. If an up-to-date model is required, the model needs editing, and the work consumes even more time. (Puhakka, 2017)

With a 3D engine configurator, the models for the sales material would always be up-to-date, which in turn would save time.

4.1.3 Delivery Center Vaasa

Project specific operation manuals, spare parts list, manufacturing instructions etc. for manufacturing and service. Generating and storing these with the help of a 3D engine configurator would be possible. Unique 3D engine models for projects, as-built and as-maintained Bill of Material could be stored for later needs.

DCV currently use a Teamcenter integrated application called Process Planner to create and view 3D engine models. These models are visualized in Teamcenter with the Life cycle viewer. A few problems occur with this method:

- Missing parts – Parts that should be added are not added.
- Supressing errors – Parts that are supposed to be supressed in NX are still visible in the Life cycle viewer.
- Part locations – The positioning of some parts are not correct in some cases.

These problems are solved by exporting the 3D data from NX to a local hard drive. Incorrectly positioned/supressed parts are repositioned. With this method, new problems occur:

- The exported data does not remain up-to-date.

- The data can only be accessed from the local hard drive. (Karppi, et al., 2017)

4.1.4 Design W31

When creating 3D assemblies and 2D drawings, the designer needs to create a specific "background" assembly inclusive of all the objects that need to be considered in the perspective of having the new components not colliding or interfering with the existing ones. These are built manually, which yields the risk of missing components. Creating these backgrounds is also very repetitive work.

Since a module can have many different variants, the design of new variants can sometimes be challenging. For example when routing pipes, collision with components may occur. Collisions can be spotted by building an over constrained situation of the area, explained in chapter 1.2.

Wrong positioning of components can sometimes be so subtle, that it is not possible to spot it. This has previously lead to collisions during engine assembly. (Cavressi, 2017)

A 3D engine configurator would provide following possibilities:

- Visualization of project specific engine configurations.
- Viewing engine configurations as sold for further development.
- Creating non-specific engine configurations for testing and investigating.

4.1.5 Services and the Wärtsilä Land and Sea Academy

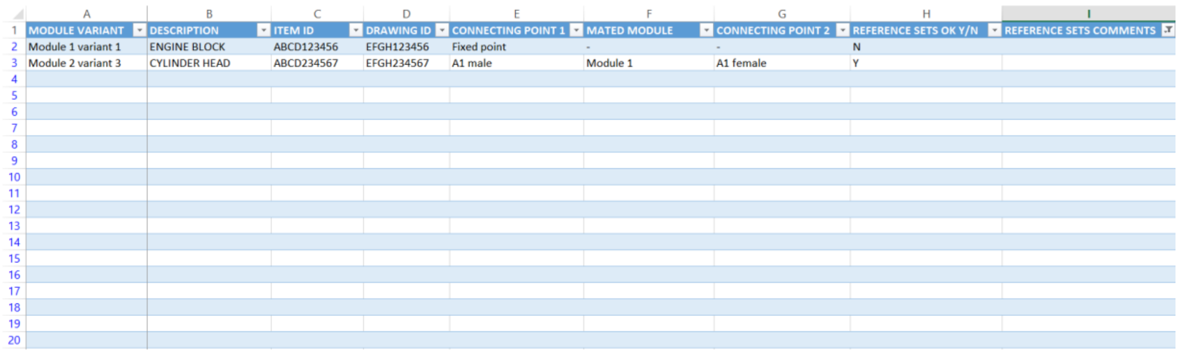
I did not interview anyone from Services or WLSA. My supervisor provided me with ideas of how a 3D configurator could be useful for WLSA. These use cases are:

- eTrainings (online training) – Creating content for online training. (WLSA)
- Service and Installation manuals (Services)

4.2 Data shape assessment

The data shape assessment was done only for the module variants used in a project specific engine. The data from the assessment are gathered in the prepared Excel spreadsheet. The Excel spreadsheet with fictional data can be seen in Figure 43. This data consists of the module number, description, item ID´s and drawing ID´s. For each module variant, there are columns with information about:

- Connecting point 1 – The name of the connecting point that is used in the module.
- Mated module – The mating module.
- Connecting point 2 – The name of the connecting point that is used in the mated module.
- Reference sets ok Y/N – Indicating if the reference sets are correct or not.
- Reference set comments – Informing what the error is and in which reference set.

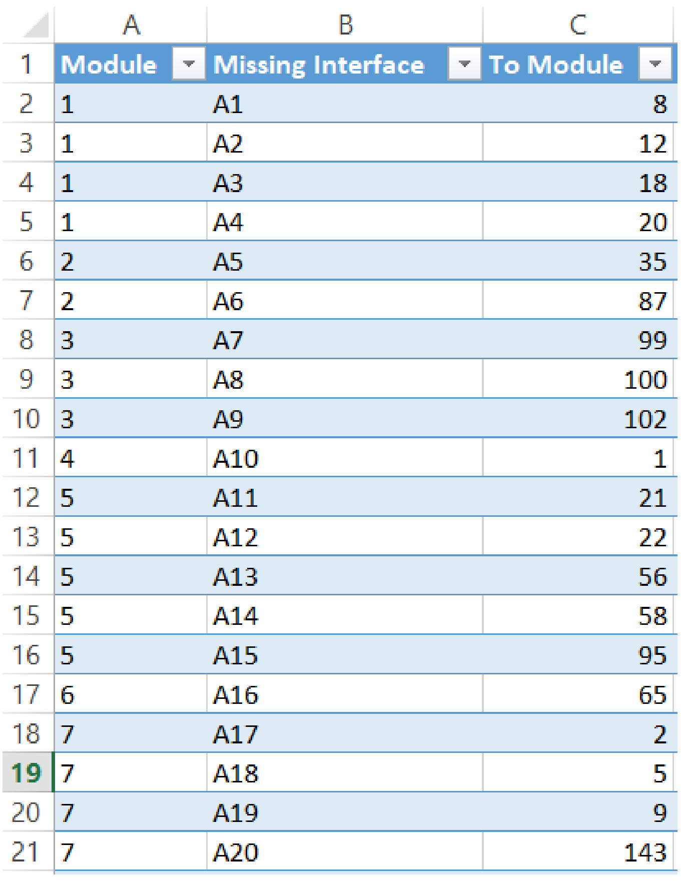


MODULE VARIANT	DESCRIPTION	ITEM ID	DRAWING ID	CONNECTING POINT 1	MATED MODULE	CONNECTING POINT 2	REFERENCE SETS OK Y/N	REFERENCE SETS COMMENTS
Module 1 variant 1	ENGINE BLOCK	ABCD123456	EFGH123456	Fixed point	-	-	N	
Module 2 variant 3	CYLINDER HEAD	ABCD234567	EFGH234567	A1 male	Module 1	A1 female	Y	

Figure 43: Excel spreadsheet from the data shape assessment

The quantity of connecting points depends on the modules. For example, one module may have five connecting points, as it mates with multiple other modules, while another module may only need one connecting point, as it only mates with a single other module.

Many of the connecting points listed in the module reports are missing from the actual models in NX. An additional spreadsheet was made for not only the project specific module variants, but also all the available variants in the standard register. In this spreadsheet, I listed the missing connecting points so that I could easier decide which ones to use to connect the module variants. This saved me a lot of time, because I did not have to go back and forth between all the modules to check if a specific connecting point is available or missing. The spreadsheet with fictional data is can be seen in Figure 44.



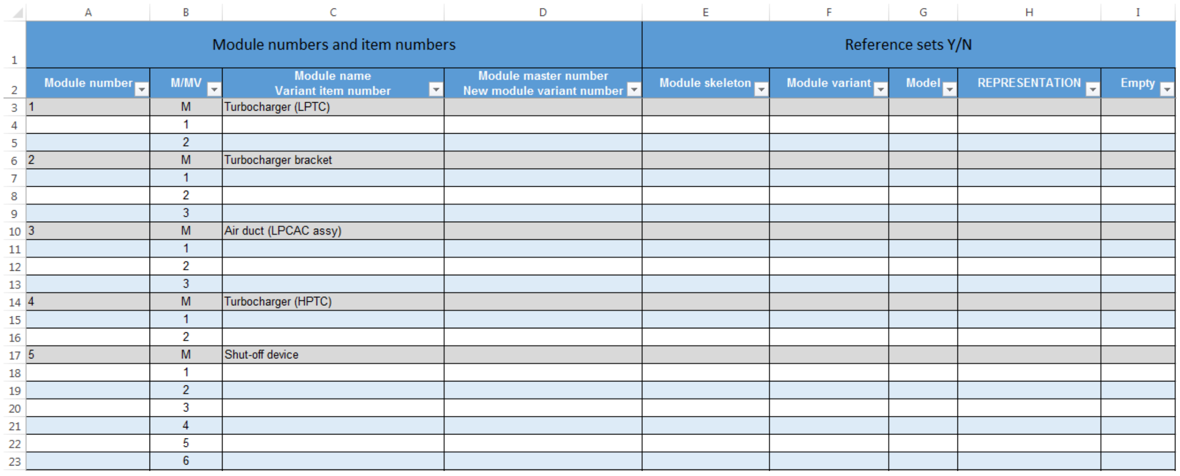
	A	B	C
1	Module	Missing Interface	To Module
2	1	A1	8
3	1	A2	12
4	1	A3	18
5	1	A4	20
6	2	A5	35
7	2	A6	87
8	3	A7	99
9	3	A8	100
10	3	A9	102
11	4	A10	1
12	5	A11	21
13	5	A12	22
14	5	A13	56
15	5	A14	58
16	5	A15	95
17	6	A16	65
18	7	A17	2
19	7	A18	5
20	7	A19	9
21	7	A20	143

Figure 44: Excel spreadsheet with missing connecting points

After the assessment, I realized that there is a need to go through all the modules and their variants and take notes about the condition of their reference set. In Figure 45, I prepared a template in an Excel spreadsheet that can be used in the future for another data shape assessment. There are five columns for each module´s reference sets. In these columns, the user can insert:

- Y – Indicates that the reference set is correctly made.
- N – Indicates that the reference set is not correctly made.

If a reference set is not correctly made, the user can also right click the field in the Excel and insert a comment. This comment should contain information about the error. It should also contain a proposal of how to correct it.



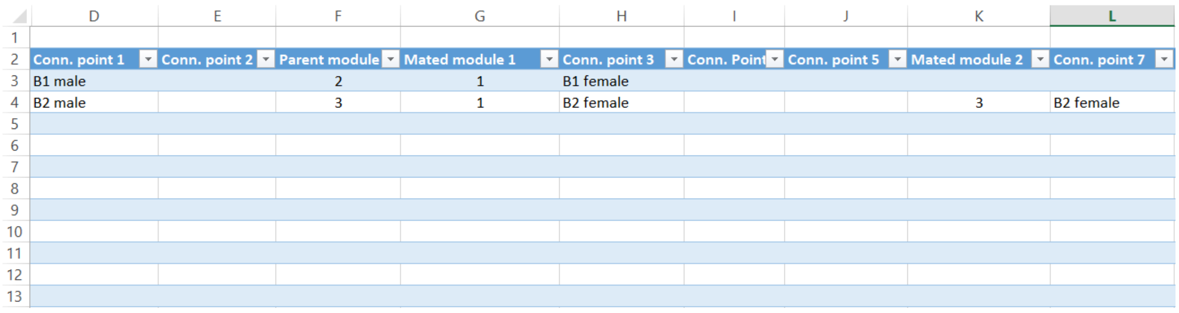
	A	B	C	D	E	F	G	H	I
1	Module numbers and item numbers				Reference sets Y/N				
2	Module number	M/MV	Module name Variant item number	Module master number New module variant number	Module skeleton	Module variant	Model	REPRESENTATION	Empty
3	1	M	Turbocharger (LPTC)						
4		1							
5		2							
6	2	M	Turbocharger bracket						
7		1							
8		2							
9		3							
10	3	M	Air duct (LPCAC assy)						
11		1							
12		2							
13		3							
14	4	M	Turbocharger (HPTC)						
15		1							
16		2							
17	5	M	Shut-off device						
18		1							
19		2							
20		3							
21		4							
22		5							
23		6							

Figure 45: Data shape assessment template

4.3 Logics

The result of the work is that approximately 80 percent of the logics for all the Wärtsilä 31 modules are completed. The logics are stored in the Rulestream Platform Server. The logics can be accessed through the Rulestream Architect and the user interface that I used for testing the logics can be accessed through the Rulestream Engineer. It is possible to modify, remove or create new logics as well as edit the user interface in the Rulestream Architect.

I used the Excel spreadsheet from the data shape assessment, in which I had listed the connecting points I wanted to use, see Figure 43. I realized that some of the connecting points had to be changed in order to get the logics to work properly. I created a new Excel spreadsheet in which I documented the new connecting points and mating modules. The spreadsheet with fictive data can be seen in Figure 46.



	D	E	F	G	H	I	J	K	L
1									
2	Conn. point 1	Conn. point 2	Parent module	Mated module 1	Conn. point 3	Conn. Point	Conn. point 5	Mated module 2	Conn. point 7
3	B1 male		2	1	B1 female				
4	B2 male		3	1	B2 female			3	B2 female
5									
6									
7									
8									
9									
10									
11									
12									
13									

Figure 46: Excel spreadsheet with logic documentation

Some of the modules have a quantity of more than one within the same engine assembly. This means that in some cases, the module might have to mate to itself. For example in

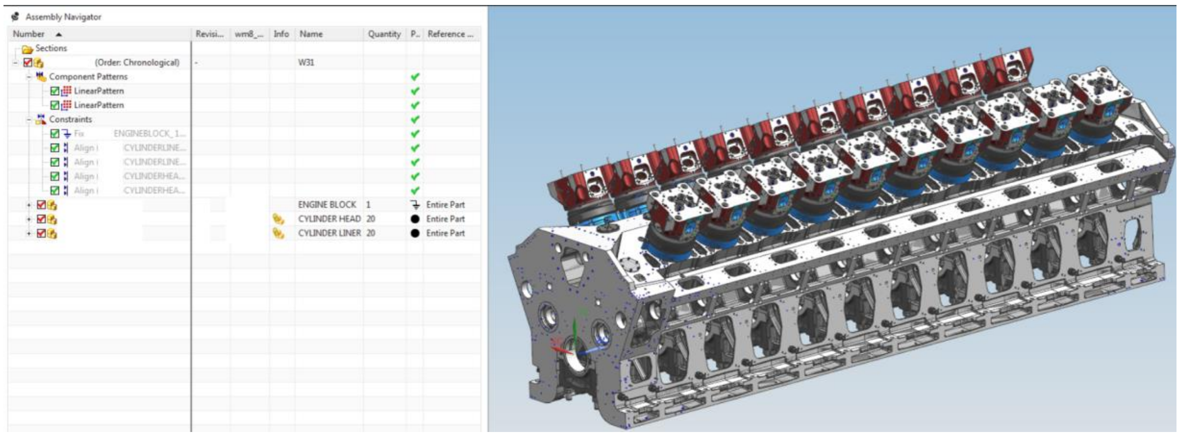
Figure 46, module 3 mates with module 1, with the connecting points B2 male and B2 female. Module 3 also mates with itself, using the B2 male and B2 female connecting points. Module 3 has both the connecting points in its model structure, and is therefore able to mate with itself.

4.3.1 Testing the logics

The logics were tested every now and then during the work process, to check if it was working as it was meant to, or if it had to be edited. In the user interface, I simply specified the engine configuration that I wanted to test, for example:

- Number of cylinders – 20
- Turbocharger location – Free end
- Technology – Diesel

In Figure 47, there is an example of the logics being tested in NX for the cylinder heads, the cylinder liners and the engine block of a W20V31 engine. Another example for a W8V31 engine is seen in Figure 48.



Assembly Navigator

Number	Revisi...	wm8_...	Info	Name	Quantity	P...	Reference ...
Sections							
(Order: Chronological)	-			W31			
Component Patterns						✓	
LinearPattern						✓	
LinearPattern						✓	
Constraints						✓	
Fix ENGINEBLOCK_1...						✓	
Align CYLINDERLINE...						✓	
Align CYLINDERLINE...						✓	
Align CYLINDERHEA...						✓	
Align CYLINDERHEA...						✓	
				ENGINE BLOCK	1		Entire Part
				CYLINDER HEAD	20	●	Entire Part
				CYLINDER LINER	20	●	Entire Part

Figure 47: Testing the logics for a W20V31 engine in Siemens NX

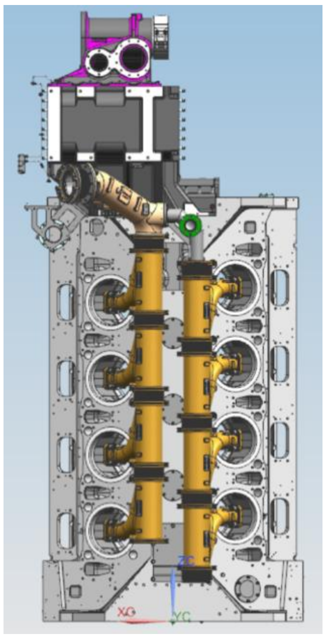


Figure 48: Testing the logics for a W8V31 engine in Siemens NX

5 Discussion

The Rulestream software I was working with was licensed with a trial license.

When the license expired, about 80 percent of the logics for all the Wärtsilä 31 modules were completed. This amount was sufficient to finalize the thesis work, without waiting for the software renewal process to complete. This decision was also taken because negotiations with software vendors take an undefined amount of time and can be a potentially long process.

The hardest part about creating the logics was to learn and understand the Visual Basic programming language. The logics for the connecting points were pretty straight forward, but the logics for the user interface and engine configurations was sometimes very challenging. Component selection and quantity is affected by the engine configuration. Some components might be depending on many different configurations, which sometimes resulted in long and difficult pieces of Visual Basic codes. Lauri Majamaa from IDEAL was very helpful and I learned a lot from him when he explained how to program Visual Basic and use the Rulestream software.

I think that once the 3D configurator is done and taken into use for daily work, it will speed up design processes a lot. The designer will not have to spend hours to search for the correct modules and their variants, for example when preparing some kind of a background assembly. This will provide faster results and make the preparation work less repetitive and less error prone. This enables the engineers to spend more of their time and skills in higher-level types of tasks, where it benefits the most to the whole company.

If I were to do the work again in the future, I would first make sure that I have a set structure of how to program Visual Basic codes for the user interface, used to test the logics. This would make it easier to go back and quickly understand the code, when changes or new codes need to be made. I changed the structure of the codes a few times during this work, so every time that I had to change something to the codes that were done earlier in the work, it was sometimes difficult, as the codes could be very long.

To continue with this work, the remaining logics are to be completed. The logics created then have to be maintained, as new module variants are created. The following step is to start testing the logics with the IOS configurator.

6 References

Cavressi, F., 2017. *Design W31* [Interview] (29 July 2017).

Ericsson, A. & Erixon, G., 1999. *Controlling Design Variants: Modular Product Platforms.* Dearborn, Michigan: Society of Manyfacturing Engineers.

Johannesson, H., Persson, J.-G. & Pettersson, D., 2013. *Produktutveckling.* Andra upplagan ed. Stockholm: Liber AB.

Karppi, L., Välimäki, M. & Kesseli, J., 2017. *Delivery Centre Vaasa* [Interview] (27 June 2017).

Puhakka, P., 2017. *Sales Material* [Interview] (12 June 2017).

Siemens PLM, 2017a. *Rulestream.* [Online]
Available at:
http://www.plm.automation.siemens.com/en/products/open/rulestream/index.shtml#lightview-close
[Accessed 2 August 2017].

Siemens PLM, 2017b. *Rulesteam Essentials,* s.l.: K. Jung, R. Smith, D. Mazure.

Södö, T., 2017. *Project Engine Room Systems* [Interview] (7 June 2017).

Wärtsilä, 2015a. *Record holder.* [Online]
Available at: https://www.wartsila.com/media/news/02-06-2015-new-wartsila-31-engine-achieves-guinness-world-records-title
[Accessed 3 August 2017].

Wärtsilä, 2015b. *The new Wärstilä 31 engine.* [Online]
Available at: https://www.wartsila.com/twentyfour7/in-detail/the-new-wartsila-31-engine
[Accessed 3 August 2017].

Wärtsilä, 2017a. *Corporate presentation 2017.* [Online]
Available at: https://www.wartsila.com/investors/reports-presentations/corporate-presentations-brochures
[Accessed 2 August 2017].

Wärtsilä, 2017b. *Wärtsilä W31 brochure.* [Online]
Available at: https://cdn.wartsila.com/docs/default-source/product-files/engines/ms-engine/brochure-o-e-w31.pdf?sfvrsn=13
[Accessed 3 August 2017].

Wikipedia, 2017a. *Engineer to order.* [Online]
Available at: https://en.wikipedia.org/wiki/Engineer_to_order
[Accessed 2 August 2017].

Wikipedia, 2017b. *Modular Design.* [Online]
Available at: https://en.wikipedia.org/wiki/Modular_design
[Accessed 4 August 2017].