

*Master Thesis*

# Capture Knowledge with Object-Process Modeling

*A systems engineering approach*

*Elinor Ahlstedt*

---

*Division of Machine Design • Department of Design Sciences  
Faculty of Engineering LTH • Lund University • 2015*



**LUND UNIVERSITY**





# **Capture Knowledge with Object-Process Modeling**

*A systems engineering approach*

*Elinor Ahlstedt*

---

*Division of Machine Design • Department of Design Sciences  
Faculty of Engineering LTH • Lund University • 2015*

Division of Machine Design, Department of Design Sciences  
Faculty of Engineering LTH, Lund University  
P.O. Box 118  
SE-221 00 Lund  
Sweden

ISRN LUTMDN/TMKT 15/5539 SE

## Preface

This report presents the work of my master thesis for a degree in Mechanical Engineering with Industrial Design. The project was carried out at Volvo Car Group in collaboration with the Division of Product Development, Faculty of Engineering LTH at Lund University.

I would like to direct a special thanks to my supervisor at Volvo Cars, Mikael Törmänen; as always with great dedication have given me guidance through the project.

I would also like to thank to Håkan Edman for given me the opportunity to write my master thesis at Volvo Cars in the first place.

I am very grateful for the thorough feedback and advice in project given by my supervisor at Lund University, Damien Motte as well as examiner Professor Robert Bjärnemo at the Division of Product Development. Furthermore, many thanks directed to Professor Johan Malmqvist at Chalmers University of Technology, who provide me with great input and literature in the beginning of the project.

It has been an utmost interesting project that has given me an opportunity for lots of new learnings and experience along the way.

Gothenburg, August 2015.

Elinor Ahlstedt



## Abstract

To increase the efficiency of the product development process at Volvo Car Group, knowledge accumulation is central in the early strategy and concept phase.

Within the department for Research and Development, the unit responsible for powertrain engineering desires a process to document system interfaces, in order to reuse what they do already know in new applications.

This thesis presents a process how to capture systems knowledge; i.e. interactions within system structure, functions and behavior with the use of object-process oriented modeling. Included in the process is also ideas presented how to manage and maintain as well as interpret and reuse captured knowledge.

During the first part of the project, literature of theory and previous empiric was explored, in order to understand principles of knowledge based development and systems engineering.

To identify needs of the desired process, system engineers responsible for the complete powertrain were interviewed. Thereafter, the interpreted needs were translated to a functional analysis of the desired process.

A case study was conducted at different developing units across Powertrain Engineering. The purpose was to map system knowledge with object-process methodology.

The result was a mapped system architecture based on the vehicle response attribute, where captured system knowledge is connected to the development phases as well as the system responsible.

The object-process oriented model of the system architecture included qualitative traceability between system requirements, decomposed functions, product structure with physical interface as well as resources defining who is owner of the system knowledge.

To illustrate how to interpret and make the captured system knowledge reusable, structural relations within three systems was mapped in a domain mapping matrix, which is a matrix mapping the dependencies between two data types. In this case requirements and functions, as well as functions and hardware components.

### **Keywords:**

*knowledge based development, systems engineering, object-process methodology, powertrain engineering; system architecture; capture, reuse*





## Sammanfattning

För att effektivisera produktutvecklingen på Volvo Car Group, är ackumulering av kunskap nödvändigt, bl.a. i den tidiga strategi och konceptfasen.

Enheten Powertrain Engineering inom avdelningen för Research & Development, önskar en process för att dokumentera gränssnitten mellan olika system på drivlinan. Detta för att minska omarbete och återanvända viktig kunskap i nya applikationer.

Det här examensarbetet presenterar en process i att fånga systemkunskap med objekt-processorienterad modellering. Idéer presenteras också för hur man ska hantera och underhålla denna kunskap samt hur man skulle kunna tolka och återanvända den fångade kunskapen.

Under den första delen av projektet genomfördes en litteraturstudie över teori och tidigare empiriska studier, i syfte att förstå principerna med kunskapsbaserad utveckling samt systems engineering.

För att identifiera behov för den önskade processen, intervjuades systemansvariga för komplett drivlina. Därefter översattes de tolkade behoven till en funktionsanalys över processen.

En fallstudie genomfördes på tvärfunktionella utvecklande avdelningar på Powertrain Engineering. Syftet var att kartlägga kunskap kring systemen med object-process methodology.

Resultatet blev en kartlagd systemarkitektur, baserat på hur drivlinan påverkar egenskapen vagnsrespons för en komplett bil. Den dokumenterade kunskapen är kopplad till de olika utvecklingsfaserna samt organisationsstrukturen och dess systemansvariga.

De objekt-processorienterade modellerna över systemarkitekturen inkluderade kvalitativ spårbarhet mellan system- och funktionella krav, funktioner, produktstruktur samt resurser integrerade i utvecklingen, dvs. den som är ägare av systemkunskapen.

En del av systemarkitekturens strukturella relationer översattes i en domain mapping matrix, dvs en matris som mappar beroenden mellan två data typer. I detta fall krav mot funktioner samt funktioner mot hårdvarukomponenter. Syftet var att illustrera hur man skulle kunna tolka den dokumenterade kunskapen i modellerna och återanvända den i nya applikationer.



## Abbreviations and Glossary

The following abbreviations are used in the report. Table 1.1 describes general abbreviations while table 1.2 describes abbreviations within Volvo Car Group.

**Table 1.1** Abbreviations used in the report.

CA	Customer Attribute
CAE	Computed Aided Engineering
DP	Design Parameter
DMM	Domain Mapping Matrix
FR	Functional Requirement
HW	Hardware
KBD	Knowledge Based Development
KM	Knowledge Management
LPD	Lean Product Development
MBSE	Model Based Systems Engineering
OPD	Object-Process Diagram
OPM	Object-Process Methodology
PLM	Product Lifecycle Management
R&D	Research and Development
SE	Systems Engineering
SW	Software

**Table 1.2** Abbreviations used within Volvo Car Group.

AL	Attribute Leader
DLI	Powertrain Installation
KD	Complete Powertrain
MU	Engine Development

NVH	Noise Vibration and Harshness
P/T	Powertrain
PSS	Product System Structure
SA	System Responsible
SDA	System Design Alternative
TU	Transmission Development
VCG	Volvo Car Group
VPDS	Volvo Product Development System

The following glossary in table 1.3 are frequent used in the report and therefore explained for better comprehension while reading the report.

**Table 1.3** Explanation of frequent used glossary.

Attribute	Within VCG, an attribute describes how the customer perceives the car with their senses, such as behaviour and performance. An attribute has a value to the customer, considered critical to quality. An attribute can be achieved with the help of one or more system solutions and/or functions.
Behaviour	In this project, behaviour refers to how processes (within a system) may transform objects and how objects could enable processes, without being transformed.
Domain	In this project, a domain refers to views in a system model. It could be the customer domain with user requirements, functional domain or the physical domain with the product structure.
Function	Within VCG, a function describe what the product does i.e. their purpose. It is a value providing process for the beneficiary i.e. the customer or driver of the car
Explicit	In this project, the term refers to knowledge stated and documented clearly, thus leaving no room for confusion.
Knowledge	Know-how for application of data and information. Understanding gained through experience, the sum of what have been perceived, discovered or learnt.  There are two types of knowledge: tacit and explicit. In this project “knowledge”, refers to systems and their structure, function, dynamic and interaction with the surroundings.

Model	A model is an abstraction of reality designed to answer specific questions. A system model is comprised of multiple views/domains. In this project model refers to a graphical representation displaying domains, communicating a process and product system structure.
Object	An object is a thing that exists or can exist physically.
Process	A process is a transformation that an object undergoes during a time. A series of actions, changes or functions bringing a result. Express the dynamics. Without processes, the objects are static.
Reuse	In this project “reuse” refers to explicit documented system knowledge for new applications, which will reduce re-work and focus development efforts.
System	A system is a set of interacting elements creating an integrated whole. A system may be decomposed via sub-systems. A system exhibits: <ul style="list-style-type: none"> <li>• Structure of objects.</li> <li>• Processes that fulfil its function.</li> <li>• Behaviour</li> <li>• Boundary to the surrounding environment.</li> <li>• Exchange of input and outputs.</li> <li>• Interface to other systems.</li> </ul>
System Architecture	The embodiment of concept, the allocation of function to elements of form, the definition of interfaces among the elements and with the surrounding context.
Tacit	In this project, the term refers to knowledge understood or implied without being stated.



# Table of Contents

<b>1 Introduction .....</b>	<b>1</b>
1.1 Volvo Car Group .....	1
1.1.1 Transformation towards knowledge based development .....	2
1.2 Problem identification .....	3
1.2.1 Desired process for capture and reuse system knowledge .....	3
1.3 Project description .....	6
1.3.1 Purpose .....	6
1.3.2 Problem formulation .....	6
1.3.3 General approach .....	6
1.3.4 Delimitations .....	6
1.3.5 Deliverables .....	6
1.4 Thesis outline .....	7
<b>2 Research approach .....</b>	<b>9</b>
2.1 Qualitative research .....	9
2.2 Applied research methods .....	9
2.2.1 Literature review .....	9
2.2.2 Interviews .....	9
2.2.3 Identify needs for desired process of capture knowledge .....	10
2.2.4 Case study .....	10
2.2.5 Object-Process oriented modeling with OPM .....	11
<b>3 Theory .....</b>	<b>13</b>
3.1 Knowledge based development .....	13
3.1.1 KM contribution to KBD .....	14
3.1.2 LPD contribution to KBD .....	14
3.1.3 SE contribution to KBD .....	15
3.2 Representing context of structural knowledge .....	16
3.2.1 Object-Process Methodology .....	17

3.2.2 Matrix methodology.....	20
3.3 Systems engineering.....	21
3.4 Requirements engineering within SE.....	22
3.5 System architecture .....	23
3.5.1 Decomposition of a system.....	23
<b>4 Powertrain systems engineering at Volvo Cars .....</b>	<b>27</b>
4.1 VPDS and organization.....	27
4.2 System Responsible (SA) .....	28
4.3 Attribute, Function and System.....	29
4.4 Requirements engineering .....	31
4.4.1 Requirement decomposition .....	32
4.4.2 Balancing process.....	34
<b>5 Need findings.....</b>	<b>35</b>
5.1 Need statements for knowledge capture .....	35
5.2 Functional analysis of interpreted needs .....	37
<b>6 Knowledge mapping of a system architecture with OPM.....</b>	<b>41</b>
6.1 Capture knowledge .....	41
6.1.1 Drafting of an object-process diagram .....	43
6.1.2 An A3 based hierarchical structure of object-process diagrams .....	46
6.1.3 Traceability.....	49
6.1.4 Nomenclature.....	49
6.1.5 Reflections from the process of capture knowledge.....	52
6.2 Manage and maintain captured knowledge .....	53
6.2.1 Support for the process.....	53
6.2.2 A3 / object-process model management.....	53
6.3 Interpret and reuse captured knowledge .....	54
6.3.1 Interpretation.....	54
6.3.2 Searchability .....	56
<b>7 Conclusions.....</b>	<b>57</b>
<b>8 References .....</b>	<b>61</b>
<b>Appendix A : Interviews .....</b>	<b>65</b>
<b>Appendix B : Questions addressed to SA1 for need findings.....</b>	<b>67</b>
<b>Appendix C : Interview guide used in case study .....</b>	<b>69</b>
<b>Appendix D : Gantt chart for thesis project .....</b>	<b>71</b>



<b>Figure 1.1</b>	Mapping of systems interfaces, own figure. ....	3
<b>Figure 1.2</b>	Object-process oriented model of desired process.....	5
<b>Figure 2.1</b>	Case study process.....	11
<b>Figure 2.2</b>	Research approach described with an object-process oriented model. ....	12
<b>Figure 3.1</b>	DIKW hierarchy .....	13
<b>Figure 3.2</b>	Knowledge value stream in LPD .....	15
<b>Figure 3.3</b>	Example of an object-process diagram with OPM. ....	19
<b>Figure 3.4</b>	Mapping of domains in axiomatic design.....	20
<b>Figure 3.5</b>	V-model of system life cycle .....	21
<b>Figure 3.6</b>	Requirement decomposition and balancing process.....	22
<b>Figure 3.7</b>	Decompositional view of a medium system, .....	23
<b>Figure 3.8</b>	Generic OPD template for Whats and Hows .....	25
<b>Figure 4.1</b>	VPDS and the system engineering activities. ....	27
<b>Figure 4.2</b>	Organization of Powertrain Engineering within VCG R&D.....	28
<b>Figure 4.3</b>	Kano model of powertrain attributes, .....	29
<b>Figure 4.4</b>	Powertrain function model .....	30
<b>Figure 4.5</b>	Requirement cascading between attribute, function and system. ....	30
<b>Figure 4.6</b>	Powertrain systems engineering with the V-model .....	31
<b>Figure 4.7</b>	Flow of communication and deliveries between SAs.....	32
<b>Figure 6.1</b>	Generalized structure for documenting knowledge in an OPD .....	42
<b>Figure 6.2</b>	Example of drafted model, with questions for decomposition .....	45
<b>Figure 6.3</b>	Hierarchical structure of OPDs.....	46
<b>Figure 6.4</b>	Eight OPDs merged to a knowledge mapped system architecture. ....	47
<b>Figure 6.5</b>	Output from one model provides input to next system model.....	48
<b>Figure 6.6</b>	Footer with meta-information.....	54
<b>Figure 6.7</b>	Traceability table between decomposed functions and HW components. ....	55
<b>Figure 6.8</b>	Traceability table between decomposed functions and requirements.....	55
<b>Figure 6.9</b>	Traceability and searchability may generate requested knowledge.....	56
<b>Figure 7.1</b>	Summary of capture, transfer and reuse system knowledge.....	59
<b>Figure 7.2</b>	Capture, manage, interpret and reuse system knowledge with an OPD ...	60



# 1 Introduction

*In this chapter, a background of Volvo Car Group (VCG) is presented and their transformation journey toward knowledge based development. Then follows a problem identification and project description with the objectives of the investigation as well as the target group. Finally, the problem definition with research questions is outlined as well as delimitations and deliverables.*

## 1.1 Volvo Car Group

Volvo Car Group, also referred to as “Volvo Cars”, is a Swedish premium car manufacturer founded in 1927. Since 2010 VCG is owned by Zhejiang Geely Holding Group of China, after acquisition from American Ford Motor Company [1].

Volvo Cars is on a major transformation journey in line with the corporate and brand strategy “Designed Around You”, which describes the customer and a human centric focus. A strategic project toward 2020 is to shorten the lead-time to 20 months from program start to production ramp up. A faster renewal of products but also products in new segments will be a key factor for reaching the long-term goal to sell 800,000 cars per year globally [2].

The department of Research & Development (R&D) is responsible for the development of Volvo Cars product portfolio. This is enabled with the use of Volvo Product Development System (VPDS). It is the cross-functional development process at VCG, aiming to develop vehicles in time with the right quality [3].

### 1.1.1 Transformation towards knowledge based development

*“New knowledge leads into new areas of exploration” [4]*

Knowledge-based development (KBD) is a general expression that relates to a set of principles, methods and tools to increase the efficiency of the product development process.

This means streamlining of the processes including resources and lead times. Fast and efficient product development enables VCG to broaden the product portfolio with reduced development times; which in turn contributes to a better competitiveness and profitability.

To reach the 20 months objectives towards year 2020, knowledge accumulation is central in the early strategy and concept phase. In this phase, it is a balancing act between business goals and technical capabilities.

Therefore a necessity is:

- Knowledge about customer and market requirements and furthermore the content of the car.
- Knowledge about product attributes and how these may be improved.
- Knowledge about options available and solutions that need to be developed outside the car program.

To increase the output from product development, the organization wants to front load the engineering effort in strategy and concept phase to learn and gain knowledge before program start. This implies increasing virtual development, simulation and verification of vehicle concept before integrating in the design phase. Efforts are believed to reduce expensive rework loops and firefighting when design is complete, close to production ramp up [5].

It also requires a major change in how to manage knowledge within the organization. How to capture, store and re-use general knowledge as well as improve communication between team members and cross-functional collaboration [5].

Currently there are many initiatives within VCG to adapt principles from KBD, using working processes with methods and tools from fields such as systems engineering (SE).

## 1.2 Problem identification

### 1.2.1 Desired process for capture and reuse system knowledge

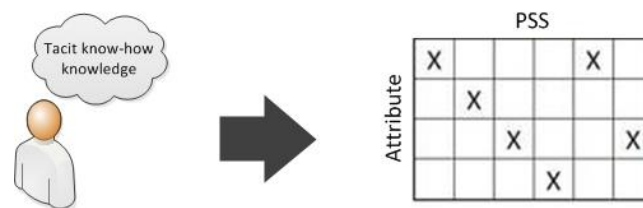
Powertrain Strategy & Concept is responsible for the Powertrain unit deliveries to the car program. This implies balanced cost-efficient concept of the powertrain between entry-level performance cars to premium high quality performance cars.

In order to deliver concepts with a certain bandwidth on the performance it is necessary to explore the design space, simulate the trade-offs between different design proposals and optimize system functions to achieve given requirements.

To facilitate the work, they need to reuse systems knowledge i.e. know-how of interactions between functions, structure and dynamics with the system environment [6], in new applications. Currently this knowledge is volatile and tacit. It exists in the minds of engineers and may disappear when team members change project, department or leave their employment, for example.

Currently, the efforts of documenting the relations are done manually. However, captured knowledge is limited to different development teams at each system level, with varying focus due to previous firefighting operations. This contributes only to a narrow view of system interfaces. The current knowledge reuse of system interactions is ad-hoc, based on carry over from previous project.

There is an ongoing work with strategies supporting documentations of systems interactions. Today the relations are described with a matrix-based method, with means to analyse and improve interfaces in the system architecture. This results in an overall mapping of the product system structure (PSS) related to attributes of a complete vehicle, figure 1.1.



**Figure 1.1** Mapping of systems interfaces, own figure.

It is a time-consuming manual work with knowledge input from several system design leaders. However, no one is responsible for maintenance so the knowledge becomes obsolete more or less, as soon the matrix has been published.

**Powertrain Strategy & Concept desires a process that:**

- Captures system knowledge by mapping the interactions between multiple domains in a system architecture.
- Is managed and maintained with a clear ownership of captured knowledge.
- Facilitates communication and shared understanding between development teams to support knowledge transfer.
- Supports systematic knowledge reuse of system interactions and structural relations in new applications.

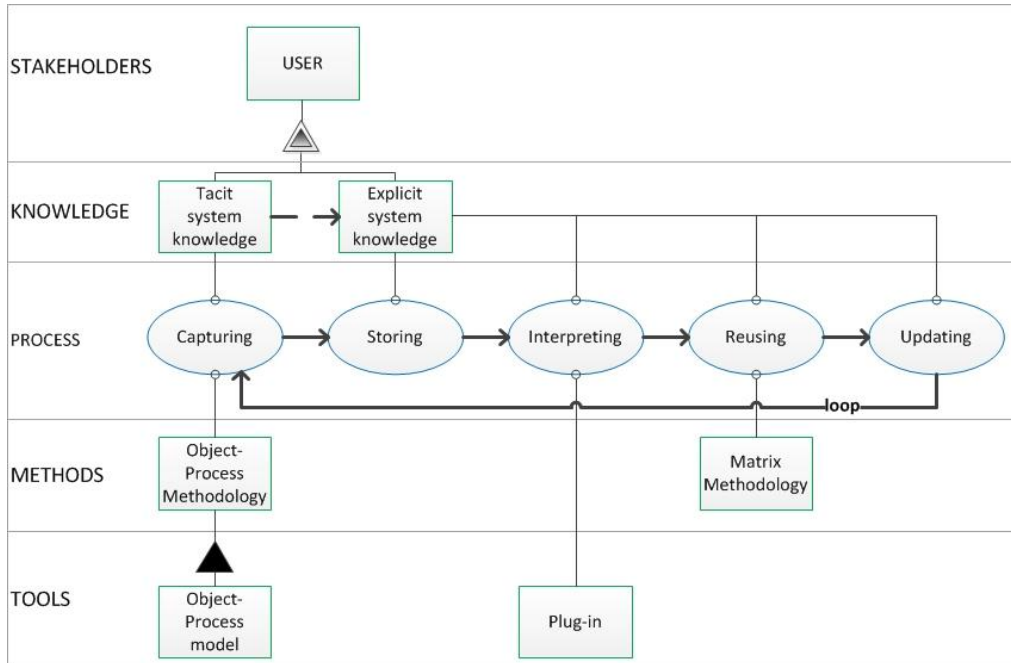
**Knowledge gap to fill:**

- How to capture tacit system knowledge and make it explicit.
- How to manage and maintain captured system knowledge.
- How to interpret and reuse captured system knowledge.

Earlier investigation made by the department of Powertrain Strategy & Concept in 2013 concluded that object-process methodology (OPM) [7] is a promising method for model-based systems engineering and knowledge mapping, but further investigation of use-cases was recommended [8].

The overall vision is that with a firm base of captured and maintained system knowledge in a model, the data could be interpreted with use of plug-in tool and matrix based methodologies to display clustered system relations. Efforts are believed to facilitate focus of development activities and use of CAE tools. In the long term, captured system interactions could provide input for better confidence between simulated and measured test data for system simulation and optimization, according to technical expert at Powertrain Engineering, VCG [9].

Figure 1.2 describes the vision of desired process with an object-process oriented model. The building blocks of OPM are described in chapter 3.2.1.



**Figure 1.2** Object-process oriented model of desired process for capture knowledge, own model.

## 1.3 Project description

### 1.3.1 Purpose

The purpose of this master thesis is to investigate a process to capture and reuse powertrain system knowledge, with the use of object-process oriented modeling and supporting methods and tools. The goal is to present conclusions and recommendations supporting the desired process.

### 1.3.2 Problem formulation

In order to ensure that the objectives will be achieved, the following research questions (RQ) were formulated:

*RQ1: What are the needs for the process of system knowledge capture and reuse?*

*RQ2: How can system knowledge be captured and reused by OPM?*

### 1.3.3 General approach

Analysis is based on theory and previous research of KBD, SE and OPM, as well as current empirical studies at VCG. The empirical studies consist of:

- A background of powertrain system engineering processes at Volvo Cars,
- Need findings for the desired process of knowledge capture with OPM.
- Case study of mapping system knowledge, captured in a model.

### 1.3.4 Delimitations

The desired capture and reuse process requires (among other things) software for creating models, plug-in and clustering algorithms that interprets data in models as well as integration with existing PLM systems. This is assumed to be available. These assumed prerequisites are therefore not investigated in current empiric study.

### 1.3.5 Deliverables

Outcome of this project is a report including:

- Described relevant theory.
- Described result, analysis and conclusions of the research questions.
- Recommendations for implementation of desired process.



## 1.4 Thesis outline

The outline of the thesis is structured around the following chapters.

### *Chapter 1 – Introduction:*

In this chapter, a background of VCG is presented and the organization's transformation journey toward knowledge based development. Thereafter a brief presentation of desired process of knowledge capture at Powertrain Engineering including knowledge gap to fill of desired process. The background is followed by a project description with the objectives. Problem definition with research questions is outlined as well as delimitations and deliverables.

### *Chapter 2 – Applied methods*

This chapter describes the research approach for the project and applied methods.

### *Chapter 3 – Theory*

This chapter describes the theoretical framework of the thesis. Initially principles for knowledge based development are explored with means to investigate methods for capture, store, and reuse knowledge. Furthermore, theory of systems engineering is described. As a support for model based systems engineering, object-process methodology is described. Comprised is a summary of the most important theory from the literature review.

### *Chapter 4 – Powertrain systems engineering at Volvo Cars*

This chapter describes the background of the powertrain system engineering process at VCG, with means to describe how and with whom the system responsible communicate as well as their deliverables within the development process.

### *Chapter 5 – Need Findings*

This chapter begins with collected need statements from the SAs, referring to the desired process for knowledge capture in object-process oriented models. The need statements are interpreted with a functional analysis of the process.

### *Chapter 6 – Case Study: Knowledge mapping of a system architecture with OPM*

This chapter describes the conducted case study and the process of mapping knowledge of a system architecture with OPM. Furthermore, the result is analyzed based on the functional analysis of interpreted needs, with focus on the ones with high importance. Included is a discussion with recommendations for an implementation of process.

### *Chapter 7 – Conclusions*

This chapter concludes the research and answering the research questions.



## **2 Research approach**

*This chapter describes the approach for the project and applied methods used in the research.*

### **2.1 Qualitative research**

The data is collected with qualitative research methods. Denzin and Lincoln [10] defines qualitative research as an approach to examine peoples experiences in detail by using a collection of empirical material and methods such as interviews, focus group, case studies, discussion, observations, visual methods etc. Figure 2.2 describes the research approach with an object-process oriented model.

### **2.2 Applied research methods**

#### *2.2.1 Literature review*

Webster and Watson [11] emphasizes that “a review of prior relevant literature is an essential feature of any academic project. An effective review creates a firm foundation for advancing knowledge.”

Ridley [12] express that the literature review contextualizing the work, describing the bigger picture that provides the background and creates the space or gap for the research.

In order to address the research questions, a literature review was conducted to facilitate theory development, identify potential methods and tools to support the desired process.

#### *2.2.2 Interviews*

Hove et al. [13] states that the purpose of using interviews in empirical studies, is to collect data that cannot be acquired using quantitative methods. Interviewing people gives an insight into their opinions, thoughts and problems.

Fontana and Frey [14] describes three types of interviews:

## 2 Research approach

---

- Structured: All respondents are asked a standard list of questions in a standard order. Reduces the risk of interviewer bias and increase the reliability.
- Un-structured: Few specific questions, free to ask and be responded in any way. Collected data may be too unorganized to be analyzed.
- Semi-structured: is a combination of un-structured and structured interview. Created around a core of standard questions, however the investigator may expand on any question in order to explore a given response in depth.

Nonetheless, interviews are a time consuming data collection method. Hove et al. experiences that required effort for an interview includes activities such as scheduling of appointments, collecting background information, preparing interview guides, discussion and meeting with the actual interview, summary writing as well as transcribing the answers of the respondents.

In this project both un-structured and semi-structured interview have been used to collect qualitative data from interviewees. A compilation of conducted interviews can be found in appendix A.

### *2.2.3 Identify needs for desired process of capture knowledge*

To identify needs for structured system knowledge, a five step process by Ulrich and Eppinger [15] was used. The steps are:

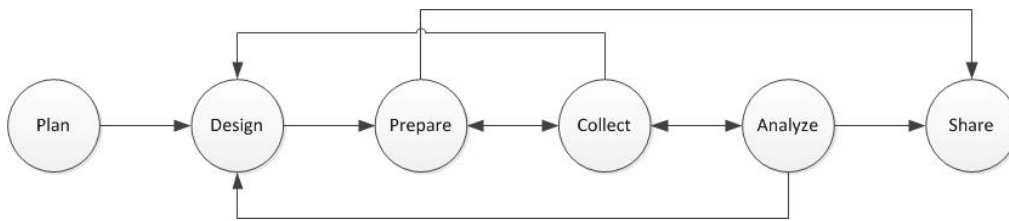
1. Gather raw data from target group.
2. Interpret the raw data in terms of needs.
3. Organize the needs into a hierarchy of primary, secondary and tertiary needs.
4. Establish the relative importance of the needs.
5. Reflect on the results and the process.

The raw data is gathered through un-structured interviews with the target group while studying their work processes. Thereafter interpreted and compiled in a list of need statements that can be found in table 5.1 in chapter 5.

The interpreted needs are translated to a functional analysis [16] of the process needed to capture knowledge. The functional analysis can be found in table 5.2 in chapter 5.

### *2.2.4 Case study*

Yin [17] points out that doing a case study is a linear but iterative process following the model illustrated in figure 2.1. Further Yin expresses that a case study can be conducted when investigating a contemporary phenomenon in depth and within its real world context. I.e. a case research is performed when you want to understand an actual case and assume that such understanding contain important contextual settings to your case.



**Figure 2.1** Case study process. Adapted from [17].

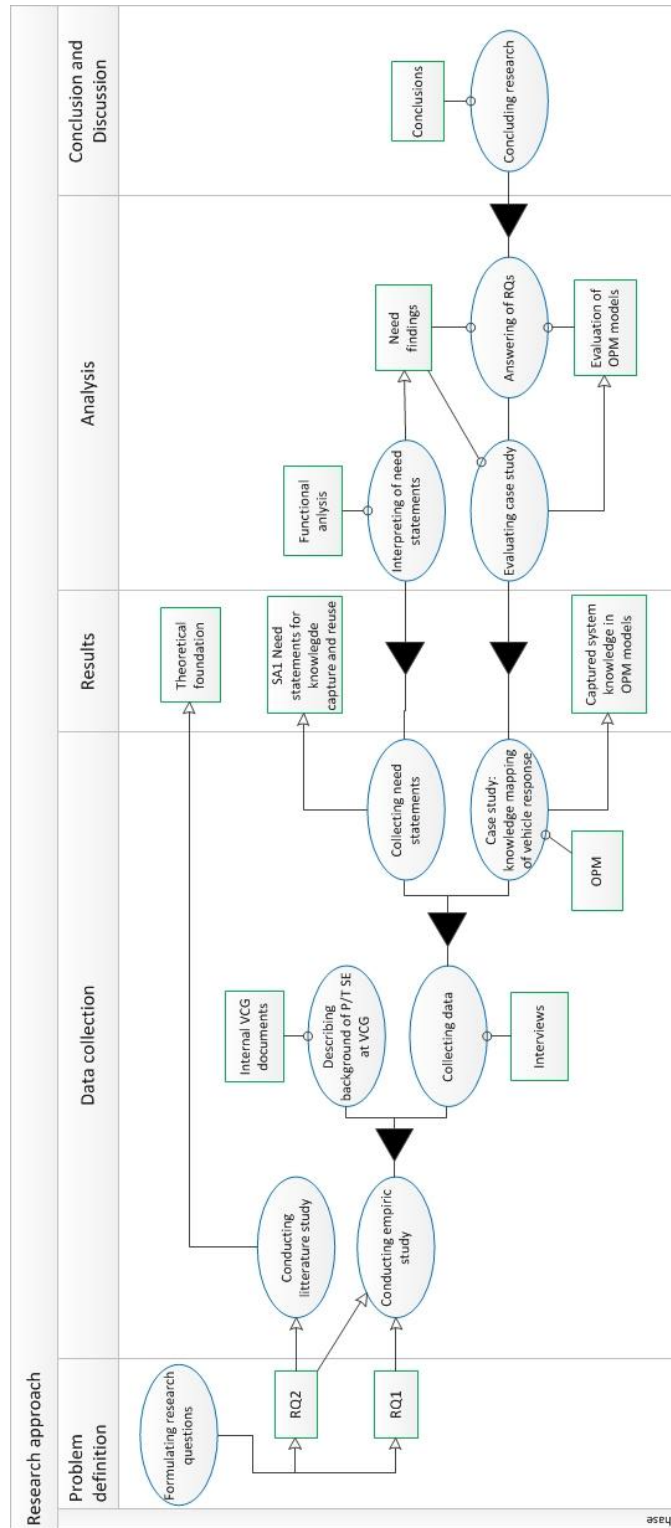
In this research, a case study has been applied in order to capture knowledge from system responsible, in object-process oriented models describing a system architecture.

### *2.2.5 Object-process oriented modeling with OPM*

For model based systems engineering object-process methodology (OPM) is a modeling language that combines a set of building blocks – objects and processes. This is modelled with a graphic-textual representation in a diagram type called object-process diagram (OPD) [7].

In this project, the data is gathered by interviews, later on categorized and stored for each interviewee, respectively, in an OPD. To draft the models Microsoft Visio was used. Templates with OPM building blocks were used, developed by VCG.

## 2 Research approach



**Figure 2.2** Research approach described with an object-process oriented model.

## 3 Theory

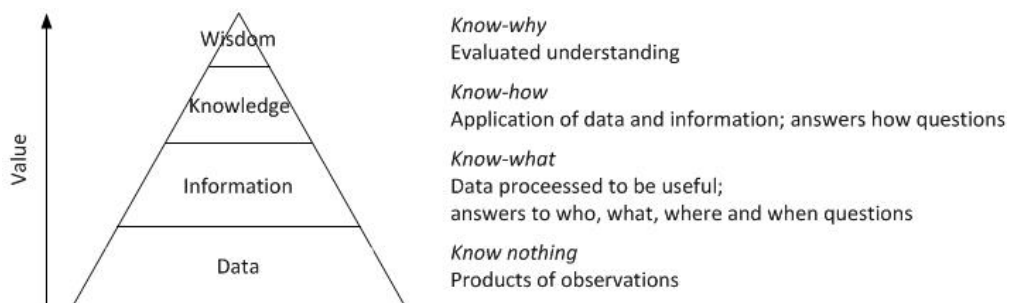
*This chapter describes the theoretical framework of the thesis. Initially principles for knowledge based development are explored with means to investigate methods for capture, store, and re-use knowledge. Furthermore, theories for systems engineering are described including processes for requirements engineering and architectural design. As a tool for model based systems engineering, principles for object-process methodology is described. Each section is a summary of the most important theory from the literature review.*

### 3.1 Knowledge based development

In an empirical study, Ulonska [18] investigated cornerstones for KBD with means to capture, store and re-use structured product knowledge.

The study concluded that KBD is a field based on processes, methodologies, tools and theories from systems engineering (SE), lean product development (LPD) and knowledge management (KM).

*What is Knowledge?* Knowledge is understanding gained through experience, the sum of what have been perceived, discovered or learnt [19]. Rowley [20] discusses the knowledge hierarchy, originated from Ackoff's paper of 1989 [21] and explaining relationship between Data-Information-Knowledge-Wisdom. The hierarchy are summarized in figure 3.1 as a pyramid.



**Figure 3.1** DIKW hierarchy adapted [20].

### 3.1.1 KM contribution to KBD

There are two types of knowledge; tacit and explicit knowledge accordingly to Nonaka and Teece [22]. Explicit knowledge is expressed in formal language and shared in forms of data, formulas and specifications that can be processed, transmitted and stored easily. Tacit knowledge is personal such as subjective insights and intuitions that are hard to formalize because it is deeply rooted in commitment, ideals and values. Tacit and explicit knowledge are complementary and both are essential for knowledge creation [23].

Nonaka & Takeuchi [24] describes the knowledge conversion within an organization based on interactions between tacit and explicit knowledge. Externalization is the mode where tacit knowledge is articulated into explicit, allowing it to be shared by others and become the basis for new knowledge.

Essential for knowledge transfer is shared understanding between individuals within a team, this includes that team members make it explicit with tools and visualizations with the use of a transfer media. To enable application and reuse of stored knowledge, it is important that the transfer media support smooth information flow between individuals within a system development team and transparency of system dependencies. This will facilitate finding the needed information and a shared understanding of the systems context [18].

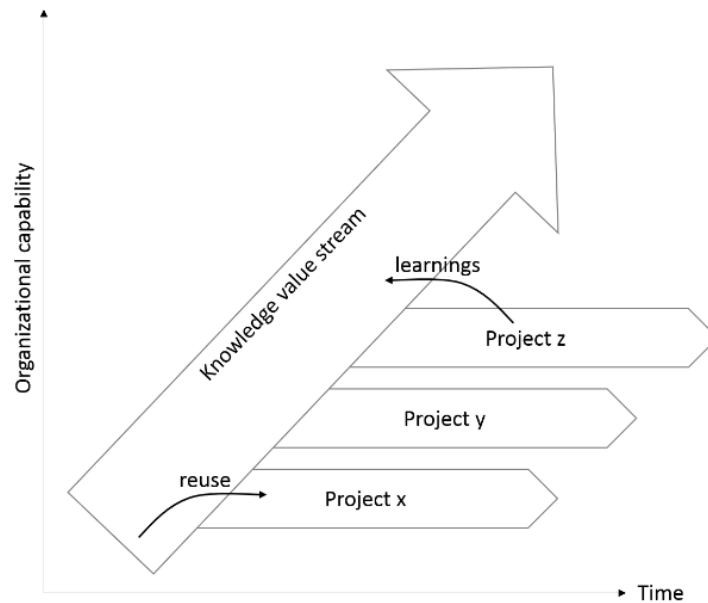
*The previous part clarifies central concepts that will recur in the report. In addition, the definitions concerning "knowledge" facilitates consistent use of terminology.*

*In this project object-process oriented models (see Section 3.3) will be investigated as a transfer media to support explicit information flow and visualize system dependencies.*

### 3.1.2 LPD contribution to KBD

Kennedy, Harmon and Minnock [25] describes that the knowledge flow across different projects representing the knowledge and product value stream within the organization, illustrated in figure 3.2. The knowledge serves as a base for project execution and integration of new learnings update current knowledge. Grant [26] views organizational capability as the outcome of knowledge integration. This means that the knowledge evolves during product development and the organizational capability increases over time.





**Figure 3.2** Knowledge value stream in LPD, adapted from [25].

Shook [27] emphasizes A3 analysis as a tool for organizational learning. An A3 is brief and precise, documenting only one problem and its solution. This makes it easy to read and grasp the context. A3 sheets that are linked in hierarchical structure can facilitate understanding of a complex problems and interrelations [18]. The A3 process for problem solving engages as well as aligns the organization by facilitating understanding and consensus of complex problem. Each person on each level in the organization have explicit responsibility and ownership. The A3 process clarifies responsibilities by placing ownership directly to the author of the A3, the one whose initials written on the paper. The structured process force team members to observe the reality and present the facts.

*The figure of the “knowledge value stream” has an utmost central role in the VCG organization and represents the change process towards a knowledge-based product development. Nor is A3 management with documented knowledge new for the organization. Previous investigations have been made to use the method to a greater extent. Therefore, it could be appropriate to adapt this method to manage documented knowledge in object-process oriented models.*

### 3.1.3 SE contribution to KBD

A complex system is managed with thorough information created in a suitable structure, in an appropriate order, and with traceability between different elements. Only the right people can see and change that information. The approach encourages teamwork with integrated product teams, yet ensures personal responsibility [28].

SE contributes KBD with methods to manage complexity by system decomposition and visual modelling [18]. SE is an enabler to solve problem of increasing complex

and multidisciplinary products, with the goal to meet user needs and reduce risk by supporting products life cycle processes [29].

*Further reading about the general systems engineering process is described in section 3.3 and theory about system decomposition in section 3.4 Background about the powertrain systems engineering process at Volvo Cars is described in chapter 4.*

### **3.2 Representing context of structural knowledge**

System knowledge includes architectural aspects, performance, quality, principal components, functionalities, concept choices and design parameters, etc. However, it is also related to the individuals in the company processing the product engineering knowledge [18].

In engineering, the context of structural knowledge may be represented with functional models and system architectures for example. An approach for this representation is with Model-Based System Engineering (MBSE) techniques [18].

Gianni, D'Ambrogio and Tolk [30] describes MBSE as a methodology which focuses on creating and using models as the primary means for information exchange between engineers. Recently focus has also covered model execution and simulation. An approach for MBSE is OPM, described in section 3.2.1.

Bruun [31] emphasizes the role of visualization in a model. A visual context makes it easier to obtain an overview of the model. A graphical representation of systems with a diagram for example is a powerful way to hold information and to share knowledge between designers and decision makers. However, a difficulty can be to present data and information visually without losing the depth of information.

Another approach for representing structural knowledge is with matrix methodologies. This is described in section 3.2.2.

Product Lifecycle Management (PLM) systems integrates management of all product related information and processes through the entire product life cycle. PLM systems supports KBD by improving knowledge access and maintenance. Furthermore, PLM serves as communication channel, making knowledge available for application and reuse [18].

*It seems that SE is also about the interactions between people i.e. resources within the organisation and the development process. Hence, it is important to integrate the role of the resources when documenting system knowledge. This will facilitate to find the owner of the system knowledge.*

### 3.2.1 Object-process methodology

System knowledge is defined in its architecture. Dori [6] claims that systems will be increasingly knowledge driven. He emphasizes the need for tools and environment for extracting existing knowledge, and generating new knowledge. Furthermore, Dori argues that a consistent and comprehensive mapping of knowledge about systems must account for their function, structure, dynamics as well as the interaction with environment, creating a collection of systems.

According to Dori [6] “object-process methodology has been developed as a holistic approach to the study and development of systems, integrating object-oriented and process-oriented models into a single frame of reference.” OPM integrates function, structure and behavior in one model. Dori [7] express that OPM is a departure from object-oriented approach when it premises that processes as entities in addition to objects. According to Dori, objects and processes are both equally important to describe a system.

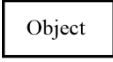

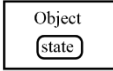



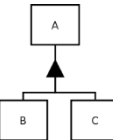
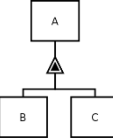
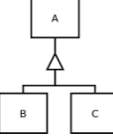

OPM is a modeling language that combines a set of building blocks – objects, which may have different states and processes that transform an object from one state to another. This is modelled with a graphical-textual representation in a single system diagram called object-process diagram (OPD).

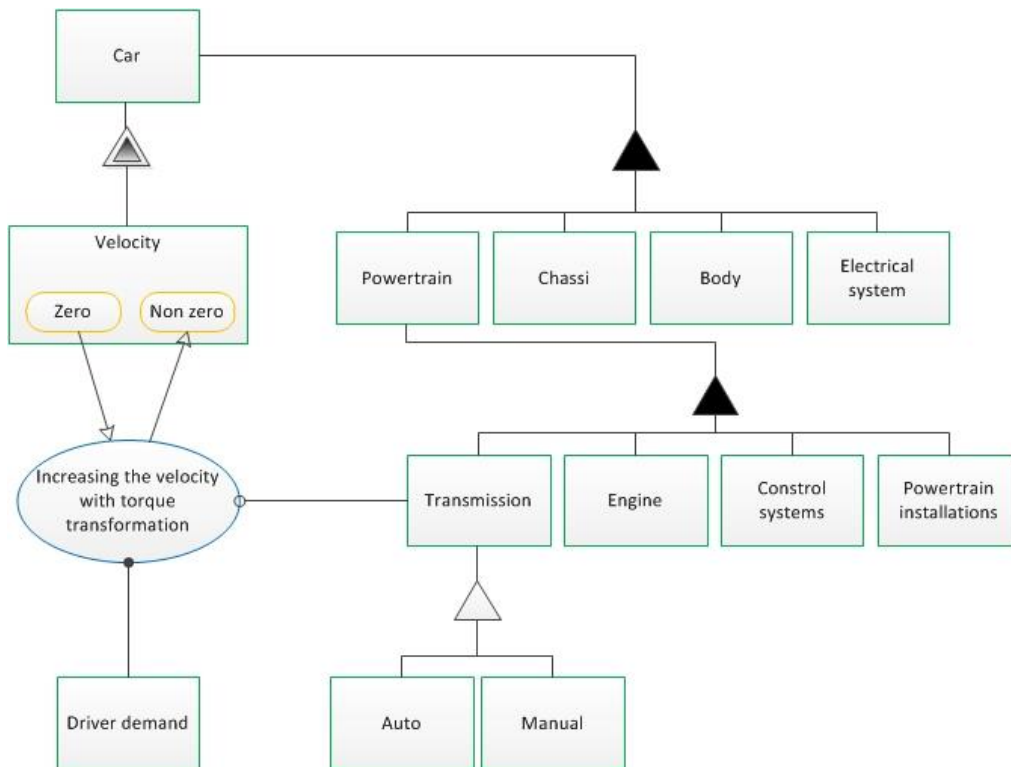
The elements of OPM are divided into three categories [7]:

- Entities: object (rectangles); states (rounded corner rectangles within objects) and processes (ellipses)
- Procedural links: Directed lines connecting processes to objects. Express transformation of the system made by process enablers.
- Structural relations: Set of triangular symbols representing fundamental structural representations.

Table 3.1 Building blocks in OPM and figure 3.3 illustrates an example of an OPD of a system with OPM.

**Table 3.1** Building blocks in OPM [7].

<i>Entities</i>	Object		An object is a static thing that exists. Can be generated, changed or consumed by processes during a time.
	Process		A process is a dynamic thing, defining how an object is transformed.
	State		A state is a situation an object can be. A process can change an object's state.
<i>Procedural links</i>	Consumption		Processing consumes object, uses entirely under its occurrence.
	Result		Processing yields object and creates an entirely new object under its occurrence.
	Input / Output		Processing changes object input state to output state. The object is at input state prior to the process occurrence and at output state as a result of its occurrence.
	Agent		Object manages processing. Object is a human, not changed by the process. Process needs human agent in order to occur.
	Instrument		Processing requires object. Object is a non-human, not changed by the process. Process needs an instrument in order to occur.
<i>Structural relations</i>	Aggregation / Participation		A consists of B and C. B and C are parts of the whole A.
	Exhibition / Characterization		A exhibits B as well as C. B and C are attributes of A.
	Generalization / Specialization		B is an A, as well as C is an A. B and C are types of A.
	Structural link		Relation between objects.



**Figure 3.3** Example of an object-process diagram with OPM. Own figure.

### 3.2.2 Matrix methodology

Börjesson's [32] study describes domain mapping matrix (DMM) as a matrix that maps the relation between two different data types (domains). Börjesson describes that DMM is an inter-domain matrix, accordingly to Malmqvist's [33] categorization of matrix-based product modeling methods. The matrix describes different types of elements in rows and columns. The relations between this elements is marked in the cells.

Axiomatic design is a system design methodology, using a matrix to analyze the transformation of customer needs into functional requirements, design parameters, and process variables according to Suh [34]. Functional requirements (FRs) are related to design parameters (DPs):

$$\begin{bmatrix} FR_1 \\ FR_2 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} DP_1 \\ DP_2 \end{bmatrix}$$

The four domains of axiomatic design are the customer domain, the functional domain, the physical domain and the process domain [35].

Customer attribute [CA]: what a customer desires from a product.

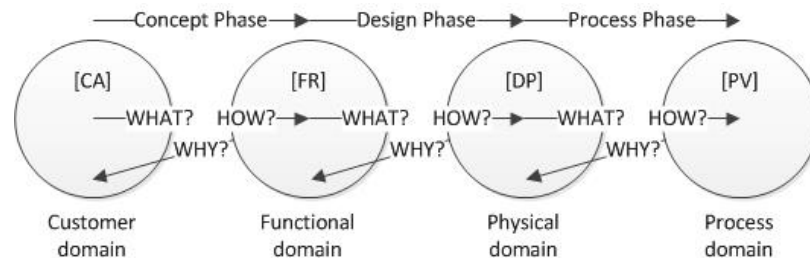
Functional requirement [FR]: minimum set of independent requirements that completely characterize the functional needs of the product in the functional domain.

Design parameter [DP]: Key physical variables in the physical domain that characterize the design that satisfies the specified FRs.

Process variables [PV]: key variables in the process domain that characterize the process that can generate the specified DPs [36].

- What should be accomplished?
- How should this be accomplished?
- Why should this be accomplished?

These are all adequate questions when moving back and forth between these domains. Figure 3.4 describes the mapping of the domains including the design phases.



**Figure 3.4** Mapping of domains in Axiomatic design. Adapted from [35].

*In this project, a matrix method is used to map dependencies between different domains expressed in an object-process oriented model.*

### 3.3 Systems engineering

*This section about processes within SE provides a basis to understand the work at Powertrain Strategy and Concept at VCG.*

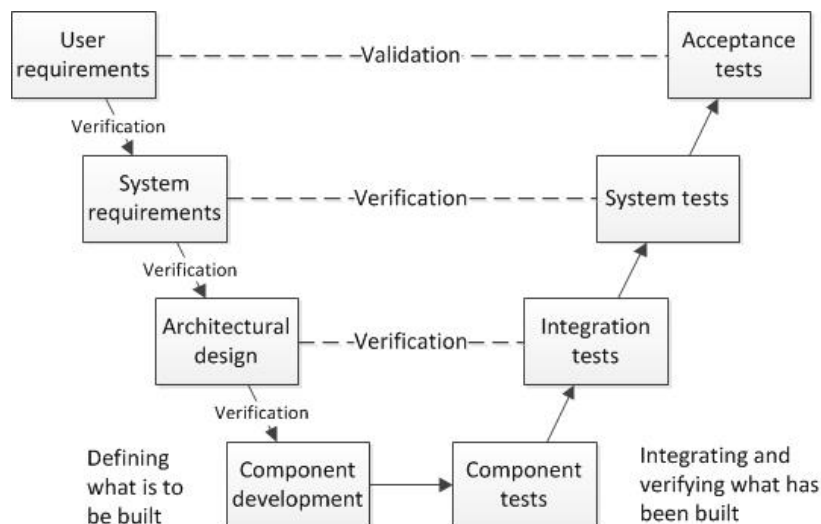
Systems engineering describes as follows:

"An interdisciplinary approach and means to enable the realization of successful systems" by INCOSE *SE handbook* [29].

"Systems engineering is a robust approach to the design, creation, and operation of systems. In simple terms, the approach consists of identification and quantification of system goals, creation of alternative system design concepts, performance of design trades, selection and implementation of the best design, verification that the design is properly built and integrated, and post-implementation assessment of how well the system meets (or met) the goals." By NASA's *Systems engineering handbook* [37].

"The systems engineering process recognizes each system is an integrated whole even though composed of diverse, specialized structures and sub-functions. It further recognizes that any system has a number of objectives and that the balance between them may differ widely from system to system. The process seeks to optimize the overall system functions according to the weighted objectives and to achieve maximum compatibility of its parts." *Systems engineering tools* by Harold Chestnut [38].

Figure 3.5 describes the system engineering process with the V-model.



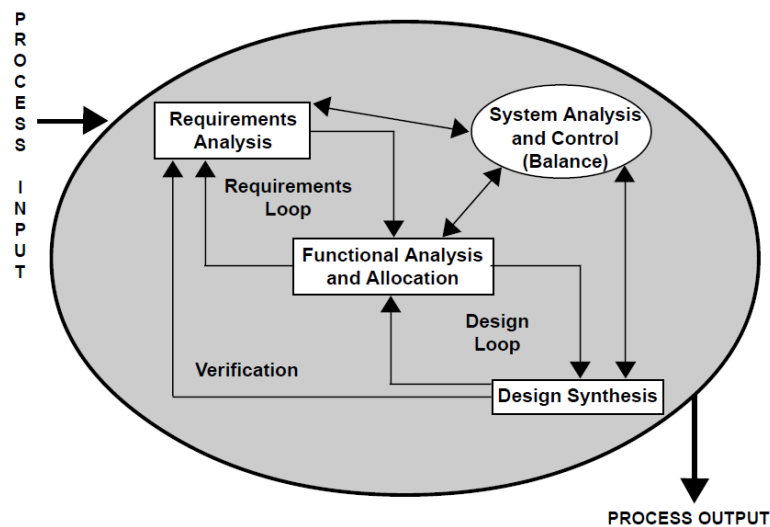
**Figure 3.5** V-model of system life cycle, adapted from [28].

### 3.4 Requirements engineering within SE

Requirements engineering is part of the SE process. In requirements engineering, traceability refers to the understanding how high-level requirements - objectives, goals, needs etc. are transformed into lower level requirements - product specification and solution. In engineering, the interest may focus on how *user requirements* are met by *system requirements* that are partitioned into *sub-system* that are implemented into *components*.

Traceability could be implemented by linking requirements from one level to another level in the design process [39]. Almfelt [40] distinguishes in his study various degrees of traceability and defined two extremes of a scale. Qualitative traceability enables identification of parts that for example are related to a requirement. These requirements can simply be met or not. Almfelt argues that quantitative traceability is a more efficient with ability to clarify how much a change or a design solution or attribute affects a certain requirement.

Figure 3.6 shows a general model of the requirement decomposition and balancing process within SE. It may present the left side of the V-model in figure 3.5.



**Figure 3.6** Requirement decomposition and balancing process. Adapted from [41].



### 3.5 System architecture

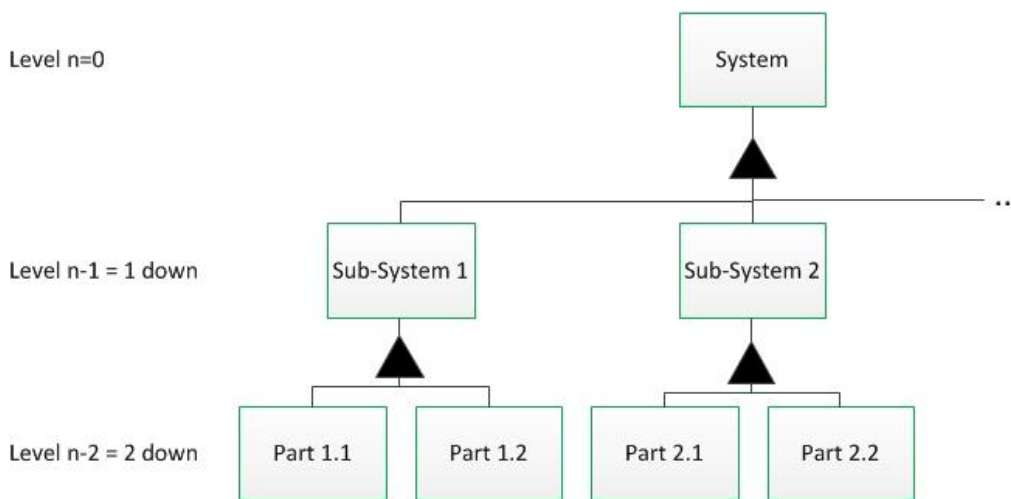
Ulrich [42] defines the essential elements of an architecture including:

- The arrangement of functional elements.
- The mapping of functional elements to physical components.
- The specification of interfaces between components.

Crawley [43] defines an architecture as: “An abstract description of the entities of a system and the relationship between those entities. “The embodiment of concept, and the allocation of physical function to elements of form, and definition of interfaces among the elements and with the surrounding context.”

Dori [7] defines the architecture of a system as the combination of the system’s structure and behavior that enables it to perform its function.

#### 3.5.1 Decomposition of a system



**Figure 3.7** Decompositional view of a medium system, adapted from [43].

The purpose of decomposition by functional analysis is to develop a system architecture. The importance of the process is to identify what the system must perform to fulfill its objectives. The three key steps in the decomposition process are:

1. Translating top level requirements into functions and derived technical requirements
2. Decompose and allocate the functions to lower levels of product system structure.
3. Identify and describe functional and sub-system interfaces.

### 3 Theory

---

The decomposition facilitates understanding of the system relations between requirements and furthermore ensures that all the requirements are allocated to at least one function. Conflicts can be identified and resolved [44].

Soderborg's [45] study describes the "What and How" decomposition as one approach for system architecture design.

- "What" refers to what is desired, an objective or requirement.
- "How" refers to how the objective or requirement is fulfilled.

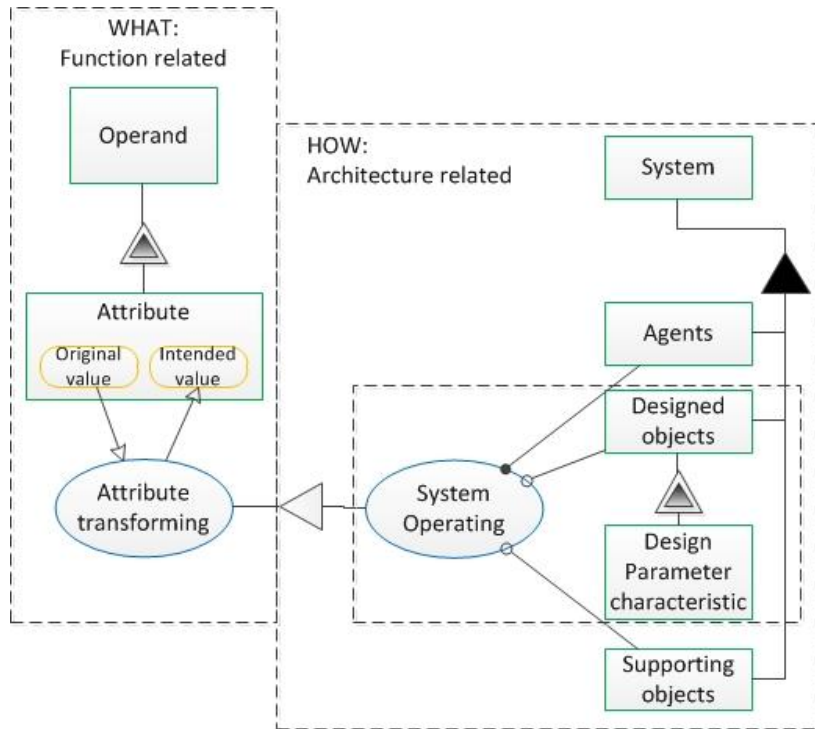
4 basic questions for good What and How decomposition of a system (46)

1. What should the system affect?
2. What effect should the system cause?
3. How does the system behave?
4. How is the system structured?

Soderborg et al [46] organizes WHATs and HOWs by OPM concepts in accordance to table 3.2. Figure 3.8 illustrates a generic OPD template for Whats and Hows.

**Table 3.2** Organizing of Whats and Hows by OPM.

WHAT? <i>What result is desired?</i>		HOW? <i>How does the system achieve it?</i>	
Function: operand-use combination		Architecture: dynamics-structure combination	
Object element	Process element	Process element	Object element
<i>What should be affected?</i>	<i>What is the desired effect?</i>	<i>How does the system operate?</i>	<i>How is the system structured?</i>
Operand-state transformee	Use, service	Behavior, operation	Structure, form



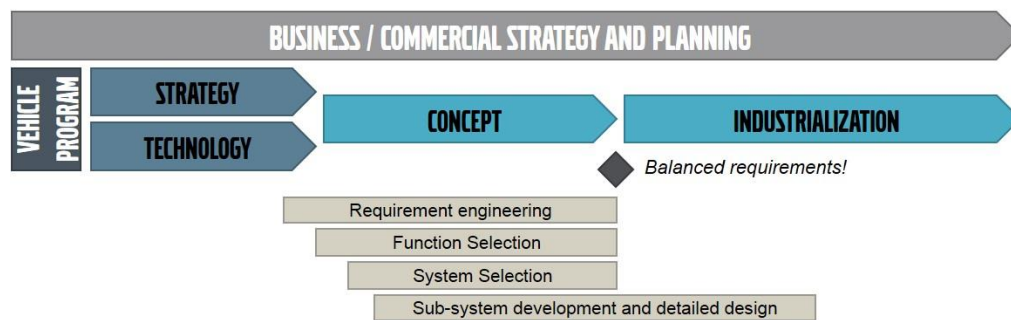
**Figure 3.8** Generic OPD template for Whats and Hows, adapted from [46].



## 4 Powertrain systems engineering at Volvo Cars

*This chapter describes the background of powertrain systems engineering process at VCG, with means to describe how and with whom the system responsible communicate with as well as their deliverables within the development process.*

### 4.1 VPDS and organization



**Figure 4.1** illustrates VPDS and the system engineering activities.

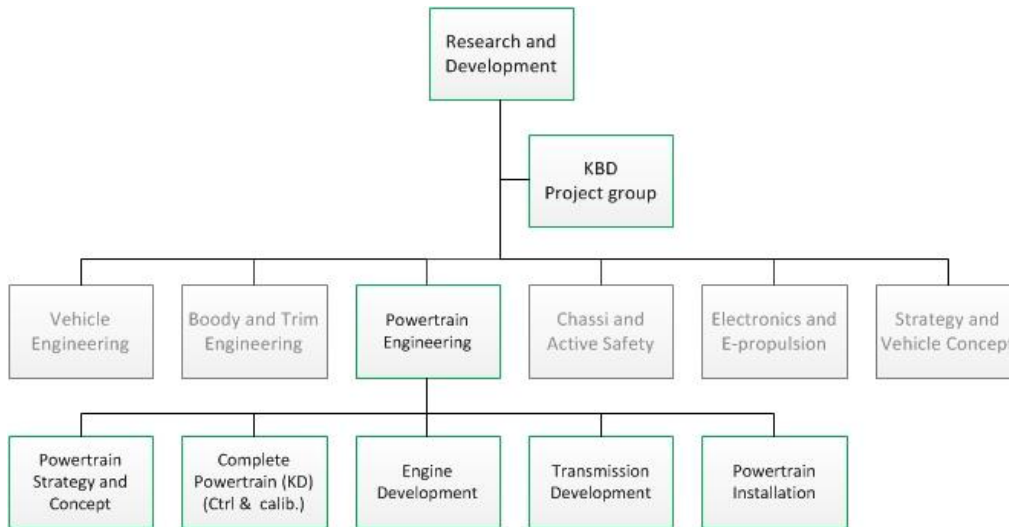
Powertrain Engineering is a unit within the department of R&D, responsible for delivering verified complete powertrains to the vehicle projects (figure 4.2). The unit consists of:

- Powertrain Strategy and Concept: responsible for powertrain attributes and unit deliveries in strategy and concept phase.
- Complete Powertrain: develop electronic hardware/software (HW/SW) control system and calibration with respect to powertrain attributes.
- Engine Development: develop diesel and petrol engines.
- Transmission Development; develop gearbox, shifter, clutch, all wheel drive system, drive shaft etc.
- Powertrain Installation: develop fuel, cooling, exhaust, air intake systems etc.

Each developing unit within Powertrain Engineering is divided into product system structure (PSS) areas. Each PSS may consists of several sub-systems.

## 4 Powertrain systems engineering at Volvo Cars

---



**Figure 4.2** Organization of Powertrain Engineering within the R&D Department.

### 4.2 System Responsible (SA)

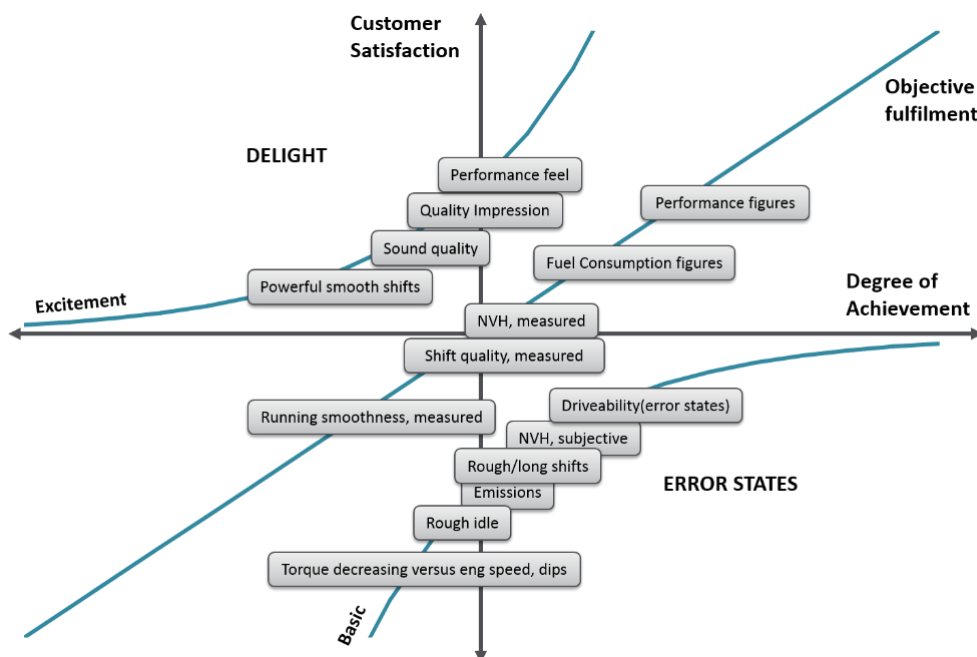
With constantly increasing demands on passenger cars, regarding emissions, fuel consumption, performance, driveability, noise, vibration and harshness (NVH), quality and cost; it is mandatory to use all available and effective methods for systems engineering in early phases to quickly find an optimal solution for the system architecture. VCG has a group of system responsible (SA) working with system engineering on different levels [47].

- **SA 0** Complete vehicle (cv)
- **SA 1** Complete powertrain (unit)
- **SA 2** Combustion engine (pss)
- **SA 3** Combustion system (sub-system)

### 4.3 Attribute, Function and System

Following is an explanation of an *attribute*, a *system* and a *function* at VCG.

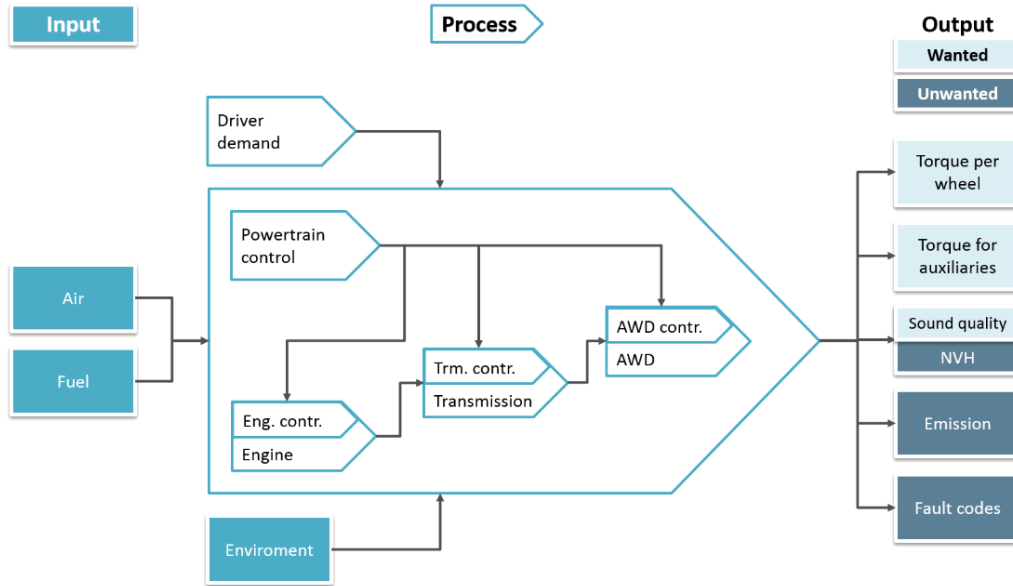
According to VCG business management system [48], an *attribute* describes how the customer perceives the car with their senses, such as composition, behavior and performance. An attribute can be achieved with the help of one or more system solutions and/or functions. Attributes has a special value to the customer and considered as critical to quality. An attribute is managed by an attribute leader [48]. Figure 4.3 illustrates a Kano model of powertrain attributes [47].



**Figure 4.3** Kano model of powertrain attributes, by technical specialist, VCG.

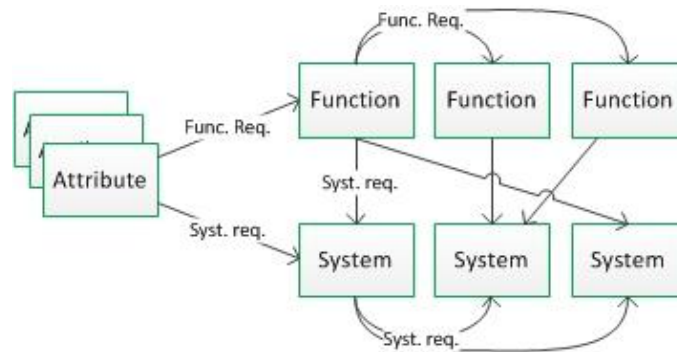
A *function* describes something that the product does. There are two types of functions: customer function, used by the customer and base function, needed by the vehicle but not always apparent for the customer. A function shall be decomposed into function partitions. Each partition represents a logical step to the function solution. A function can be achieved by support from one or more PSSs [48]. Figure 4.4 illustrates a functional model of a powertrain [47].

There are many definitions used for a *system*. Within the requirement processes and system development, a system is defined as a cluster of complete components – hardware and software, which only can be attached to one PSS. A PSS is managed by a SA2 [48].



**Figure 4.4** Powertrain function model, by technical specialist at VCG.

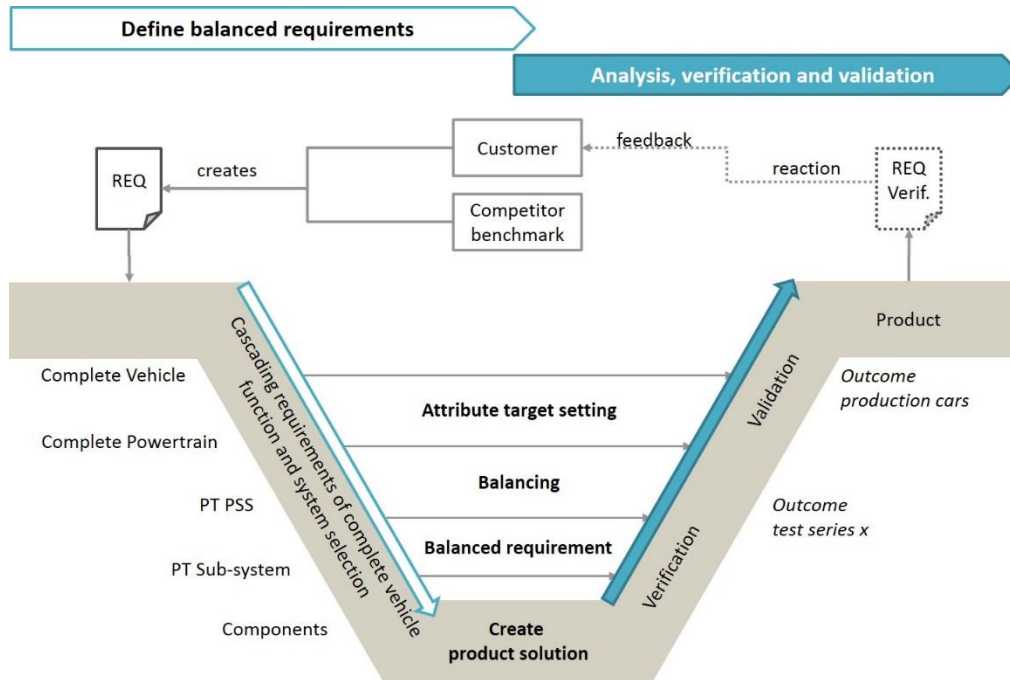
Figure 4.5 describes the relation between attribute, functions and systems. Attribute requirements for a complete vehicle are customer requirements. These are derived as technical and quantified functional requirements. At a system level these requirements are derived so that the system can fulfill its part in the complete vehicle within given restrictions and contribute to the entire customer experience [48].



**Figure 4.5** Requirement cascading between attribute, function and system.



#### 4.4 Requirements engineering



**Figure 4.6** Powertrain systems engineering with the V-model [48].

Main goal with the requirements engineering process is to measure and secure that each vehicle program will meet their requirements and target levels [48].

This means that a SA manages and secures deliveries for one system area before industrialization where each attribute target of quality, cost, weight and functional performance is aligned and balanced [48].

At the department for Powertrain Strategy and Concept, SA1 is responsible for presenting alternatives for set based design. This implies balanced cost-efficient concepts of the powertrain between entry-level performance cars to premium high-quality performance cars. The work of SA1 involves a continuous communication between attribute- and function leaders, to ensure how changes in prerequisites affect the performance and cost of the concept.

In order to deliver concepts with a certain bandwidth on the performance it is necessary to explore the different system configurations and their design space, simulate the trade-off between different design proposals and optimize system to achieve given requirements.

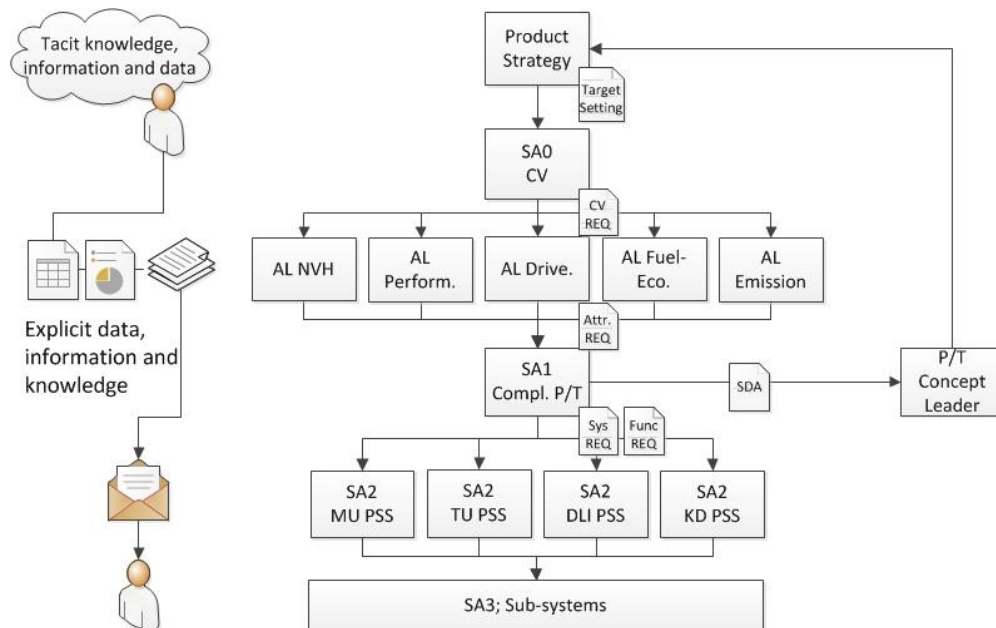
The main task of the SA1 is to identify all requirements within the system area and by using systems engineering methods, develop a complete powertrain system, which works as a whole with the underlying product system structure (PSS) areas [48].

#### 4.4.1 Requirement decomposition

Figure 4.7 mapping the flow of communication of requirements between different roles in the process. Briefly, it begins with customer needs and a benchmarking of competitors performance. Then Product Strategy decide within which attributes the Volvo brand should be leading, be among leaders, be competitive and un-competitive i.e. non-dimensioning requirements, according to SA1 at Powertrain Strategy & Concept.

For example, the safety and fuel-economy are attributes strongly coupled to the Volvo core values, therefore it is required to be among leaders. The top speed performance is not as important which results that the attribute is weighted as among leaders, according to SA1 at PT Strategy & Concept.

Product Strategy is responsible and delivers target specifications for a complete vehicle to SA0. The SA0 cascading the functional requirements to the managers of attributes effecting the powertrain. The Attribute leaders quantifies the targets and deliver both functional and system requirement to the SA1 of complete powertrain. Some requirements is directly passed on to the PSS areas [48].



**Figure 4.7** Flow of communication and deliveries. Own model.

According to SA1 at PT Strategy & Concept, system status is currently communicated through trade-off graphs in PowerPoints or data summarized in Excel, continuously updated. The communication is only stored at individuals emails or at the intranets SharePoint. The communication does not allow easy search- or traceability, neither is it indicating the relevance of the information.

There are two sets of requirements. The base requirements are always the same for a powertrain thus should not have to be structured, documented and cascaded once more for each new project. Project requirements, which are unique, always cascade for a new project and accounts for 10-30% of total. Subjective requirements are traditionally verified by driving the cars and objective requirements are measured on various system levels [47].

SA1 explain that it is impossible to optimize everything in a powertrain project. Due to the complexity, typically, a subset of the system is changed at time and the rest is carry over from a previous project.

Table 4.1 summarizes the decomposition of requirements from product planning with user requirement of complete vehicle, to functional- and system requirement for underlying subsystems within powertrain engineering. It also points out which unit is responsible for deliveries and what SA level is managing the process.

**Table 4.1** Decomposition of requirements and deliverables in PT SE process. Own summary.

SA	Unit	Responsibilities and Deliverables	SE phase
	Product Strategy	Brand- and attribute strategy targets to SA0.	Product planning
SA0	Complete vehicle	Attribute requirements to SA1 via attribute leader for powertrain attributes: <ul style="list-style-type: none"> <li>• NVH: ex sound quality,</li> <li>• Performance: ex effect engine</li> <li>• Driveability: ex acc. take off 0-100 km/h</li> <li>• Emissions: ex g KNOX / km</li> <li>• Fuel-Economy: ex g CO<sub>2</sub> / km</li> </ul>	Req. setting
SA1	Complete powertrain	Quantified functional requirements cascaded to SA2 for PSS areas. System Design Alternatives to the concept leader	System architecture Function and system selection
SA2	Product system structure (PSS)	System requirement cascaded to SA3 within: MU, TU, DLI and KD.	Balancing req.
SA3	Sub-system/	System and functional description back to SA2.	Principal solution
SA4	Components		Detailed design

#### 4.4.2 Balancing process

According to SA1 at PT Strategy and Concept [49], balancing refers to the management of vehicle attributes of a system design alternative, a concept, in order to provide user value in a cost, weight and performance efficient way.

SA1 is responsible for the system optimization for complete powertrain and multi domain optimization with the most coupled attributes. It is a complex task since there are many degrees of freedom when several systems interact with multiple attributes [47].

According to Almfelt [40], requirements are not independent. Conflicts between requirements do exist due to the complexity of a powertrain system. Trade-offs are likely to emerge during the design process. A balanced development project means that all the internal requirements conflicts are resolved, which implies a compromised and well-balanced product. However during a project many sub-systems and components are developed or changed, considered in an isolated context not interrelations in an overall powertrain system. Since requirements often are incomplete, efforts with sub-system optimization is needed.

Cost, weight and performance are the most important attributes according to SA1 at PT Strategy and Concept [49]. Performance refers to functions of the complete vehicle and their quality perceived by the user. Cost concerns the economic resources used to provide the functions. Weight refers to the total weight of the car, which should be as low as possible for best resulting fuel economy. Table 4.2 shows an example of attribute balancing, explained by SA1.

**Table 4.2** Balancing turbo configurations.

Configuration	Attributes				
	<i>Cost</i>	<i>Perform.</i>	<i>Driveab.</i>	<i>Fuel-eco.</i>	<i>NVH</i>
<i>(Set of possible design solutions)</i>	(SEK)	Effect (hk)	Response (sec)	CO <sub>2</sub> (g/km)	Sound, vibration. (dB, Hz)
Small turbo	0	0	0	0	0
Large turbo	-	+	-	-	-
Large turbo + mech. compressor	-	+	+	+	-
Large turbo + e-compressor	-	+	+	+	+

## 5 Need findings

*This chapter begins with collected need statements from SA1, referring to the desired process for knowledge capture in object-process oriented models. Thereafter, the need statements are interpreted and translated to a functional analysis of the desired process.*

### 5.1 Need statements for knowledge capture

Raw data was gathered through un-structured interviews (appendix A) during three occasions from SA1 at Powertrain Strategy & Concept. Questions addressed to the target group can be found in appendix B.

The need statements compiled in table 5.1 were noted during the interview.

**Table 5.1** Need statements from SA1.

	Need	Statement
<i>Knowledge transfer</i>	1	Be able to easily communicate a system between SAs.
	2	Be able to easily transfer their knowledge to new team members or stakeholders.
	3	Easily determine which relevant responsible stakeholders in the SE process should be convened for issues related to balancing attributes.
<i>Management / maintenance</i>	4	Be able to store model.
	5	Be able to update model.
	6	Be able to reuse model.
	7	Have a generic model of a system.
	8	Be confident that the information in the model is correct.
<i>Purpose</i>	9	Display different views of the same system such as requirements, functions, behavior and layout.
	10	A model that displays a system based on either customer- or legal requirement or corporate business level brand and strategy attributes.
<i>Balancing</i>	11	A "tick in the box", a checklist pointing at where in the

## 5 Need findings

		system calibration is needed and in what order operations need to be performed.
<i>Traceability</i>	12	Know how much something in the system affects or is affected.
	13	Know what affects the most in the system
	14	Know where changes hit further up or down the system architecture.
	15	Know how changes hit further up or down the system architecture.
	16	Know which SDA exists.
<i>Searchability</i>	17	Be able to search for a specific purpose among different models. Ex: search for "improve vehicle response" or "reduce booming noise"
<i>Notify</i>	18	Know when changes have been made in the system model.
<i>Display</i>	19	Be able to easily orient themselves in the system model.
	20	Highlight features in the model that affects or are affected in the system.
	21	Know if there is documents, reports, patents etc. that support link between requirement and function.
	22	Know where in the system critical trade-off between design proposals exists.
	23	Select how much is displayed in a model: adjacent systems, subsystems and components.
<i>Drafting</i>	24	It is important to be consistent when creating the system model.
	25	Know how a system model is quickly an easy drafted.
<i>Interpret</i>	26	A plug-in must be able to interpret the data in the model to generate new information.
<i>Value</i>	27	The output from the model is bigger than the required effort with input.

## 5.2 Functional analysis of interpreted needs

The need statements were translated in to a functional analysis [16] of the desired process of knowledge capture. The relative importance was established, on a scale of 1 to 3, where:

1. Function would be great but not necessary.
2. Function is highly desirable.
3. Function is critical.

**Table 5.2** Functional analysis of interpreted needs.

	Need	Function			Imp.
		Verb	Noun	Note	
<i>Knowledge transfer</i>	1	enables	easily communication	with stakeholders	3
	2	enables	knowledge transfer	with stakeholders	3
	2	facilitate	shared understanding of system	between stakeholders	2
	3	admits	listing	of stakeholders	2
	3	admits	structure to connect	each system level	3
<i>Management/Maintenance</i>	4	enables	storing	of model	3
	5	allows	updates	of model	3
	6	displays	current knowledge	in model	3
	7	enables	reuse of data, information and knowledge	in model	3
	8	enables	ownership	of model	3
<i>Purpose</i>	9	displays	multi domain views	in model	2
	10	provides	need-based views	in model	2
<i>Balancing</i>	11	highlights	balancing situations of requirements	in model	1
	11	enables	compilation of calibration activities	at each system level	1
	11	enables	sequential steps of calibration activities	at each system level	1
<i>Traceability</i>	12	displays	dependency weigh-	in model	1

## 5 Need findings

			factor		
	13	highlights	feature with most dependency	in model	1
	14	links	requirements to function to HW/SW	in model	3
	14	alerts	owner of model when last link is deleted	between systems	2
	14	enables	qualitative traceability	inter/intra model	3
	15	enables	quantitative traceability	inter/intra model	2
	16	displaying	SDA	in model,	2
<i>Searchability</i>	17	enables	searchability	in supporting tool for managing models	3
<i>Notify</i>	18	notifies	responsible stakeholders when changes has been made	in model, system	2
<i>Display</i>	19; 20	facilitate	traceability between dependencies	in model	2
	21	links	relevant documents	in model	1
	22	highlights	existence of trade-offs	in model	1
	23	offer	option to choose how many levels of dependencies to display	of the system architecture	2
<i>Draft</i>	24; 26	uses	equal nomenclature of element	in model	3
	24	defining	name space for elements	in model	3
	24	facilitate	drafting/development	of model	1
	24	admits	a structure/format	for the model	2
	25	admits	clear approach for drafting	of model	2
<i>Interpret</i>	26	enables	interpreting of data	in model	3
<i>Value</i>	27	offer	added value	to stakeholder	3







## **6 Knowledge mapping of a system architecture with OPM.**

*This chapter describes the case study and the process of mapping knowledge of a system architecture with OPM. Furthermore, the result is analyzed based on the functional analysis of interpreted needs.*

*Included is a discussion with recommendations for the process of capturing knowledge, managing and maintaining captured knowledge as well as interpreting and reusing captured knowledge.*

### **6.1 Capture knowledge**

In order to investigate how to map knowledge from multiple system levels, with input from several SAs, a case study was conducted.

The knowledge mapping visualizes a system architecture based on the overall customer attribute vehicle response. The result shows how the attribute is affected by the powertrain and how that affects the driveability of the car.

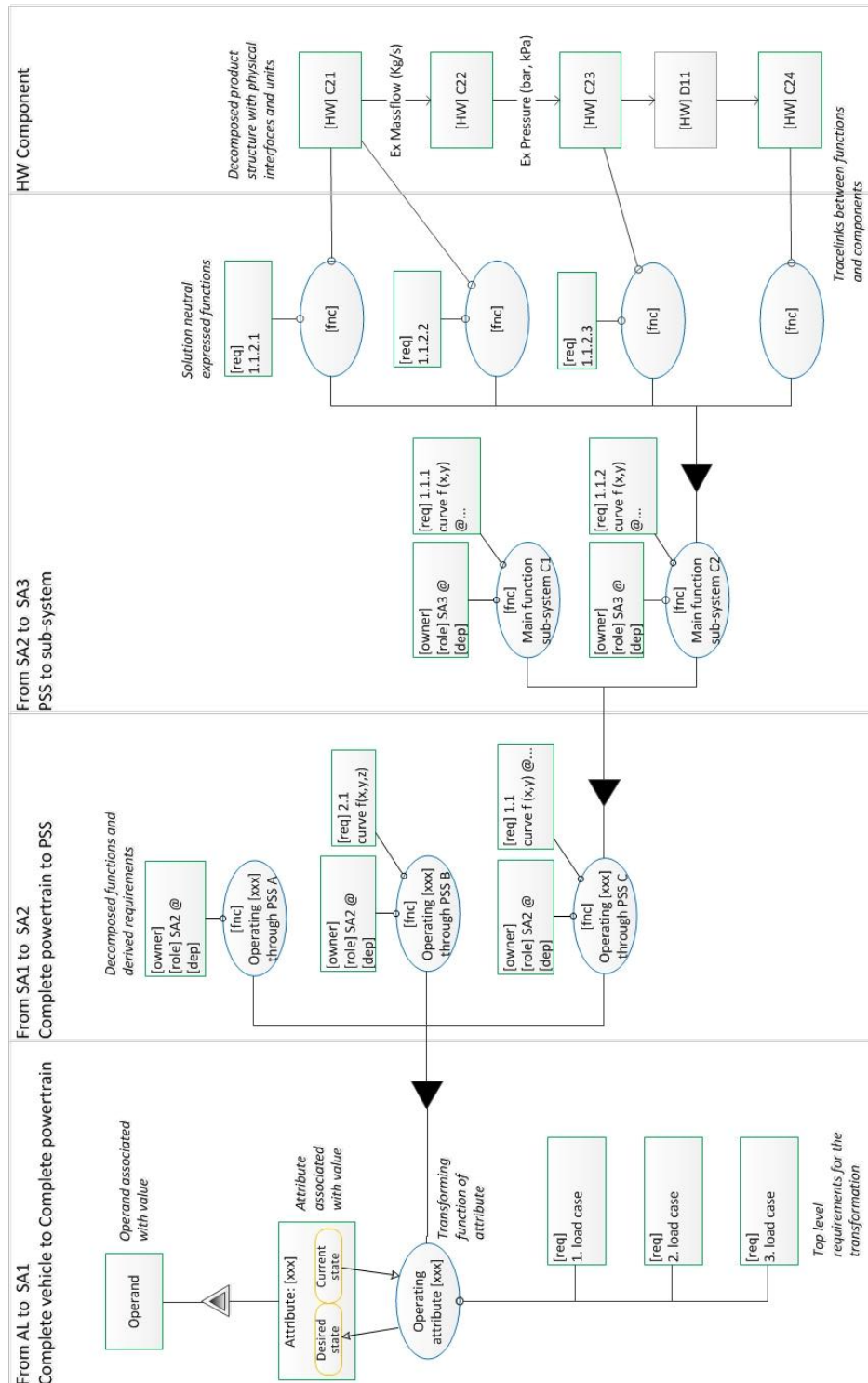
Data was gathered through semi-structured interviews from eight SAs (compiled list of interviewees can be found in appendix A), operating across the different developing units at Powertrain Engineering, at multiple system levels.

Each interview was documented and analysed in order to create a model (an OPD) of the system with OPM, at an A3 sheet. In total, nine models were created. A generalized structure of an OPD, including models from three system levels is shown in figure 6.1. An example is shown in figure 6.2.

The structure of the model is to some extent based on the outcome from previous investigation made by PT Strategy & Concept in 2013.

Before the interview with the system responsible, questions were prepared and adapted to the specific system observed. A theoretical background of the current system was also studied to facilitate the understanding of discussion in the upcoming interview.

## 6 Knowledge mapping of a system architecture with OPM



**Figure 6.1** Generalized structure for documenting knowledge through design phases in an object-process model.

### 6.1.1 Drafting of an object-process diagram

The questions in the interviews were based on Soderborg's template (see table 3.2 in chapter 3) for "What and How" decomposition of a system architecture, however, the questions differed depending on SA level interviewed. Compiled interview guide can be found in appendix C. Answers were noted directly in the interview guide.

Example of the What and How decomposition: From SA1 complete powertrain to SA2 Engine HW system, to SA3 Turbo system, see related model in figure 6.2. It describes the communication and knowledge transfer between three SAs at three system level design.

- What should the system affect? (Static, object related)
  - The driver (operand) experience of the vehicle response.
- What transformation should the system cause? (dynamic, process related)
  - Changing the vehicle response from un-acceptable to acceptable (attribute with two states).
- How does the system behave? (dynamic, process related)
  - Operating vehicle response → operating engine response → supplying boost pressure/charged air flow to engine → reducing turbo lag → reducing inertia of rotating parts in turbo system → Translating exhaust flow to rotational movement (decomposed functions).
- How is the system structured? (static, object related)
  - Turbine (component, part of the product structure)

Included in the interviews were also questions expressed in terms of:

- Why does these functions exist, i.e. which derived requirements are related to the decomposed functions?
- Who is the resource delivering or receiving the requirements?
  - Who is the owner of system knowledge? From which developing unit?
- Which are the system design alternatives?
  - Other configurations with different performances?

The discussion during the interviews was made easier by a schematic illustration of the system. This illustration was used to point at essential components in the hardware structure of the system, as well as to discuss their function and derived requirements.

Approximate time required to create a model of a system took less than an hour at a SA3 level, depending on accuracy of decomposition, i.e. how many elements with objects and processes that describes the system. However, it required less time to model the system at a SA2 and SA1 level.

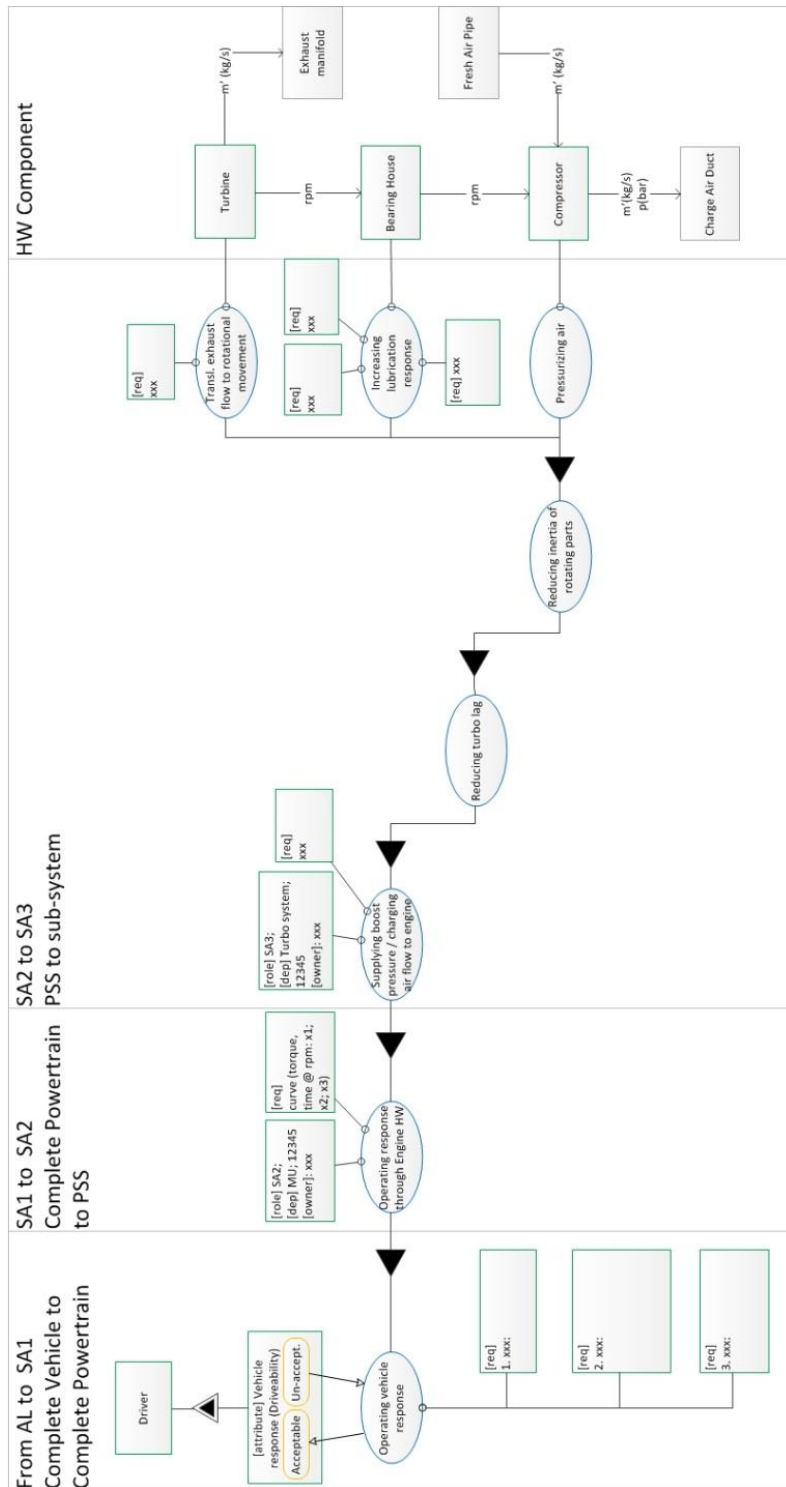
After completing the system model with an OPD, the A3 sheet was send for a review at the responsible SA for current system. This to ensure a correct content.

## 6 Knowledge mapping of a system architecture with OPM

---

During the case study, it was found that some of the mapped systems do not need further decomposition of functions and derived requirement to a SA3 level. Air intake system is a PSS area within driveline installation exemplifying this. Other PSS-areas with increased complexity requires further decomposition.

Decomposition to a SA4 level (component owner) was not carried out in the case study. During interviews number 6 and 9 it was told that many of the requirements on a detailed level design is an extra-communication between the component owner and the supplier.



**Figure 6.2** An example of drafted model, with the use of “What, How, Why, Who” – questions decomposition of system.

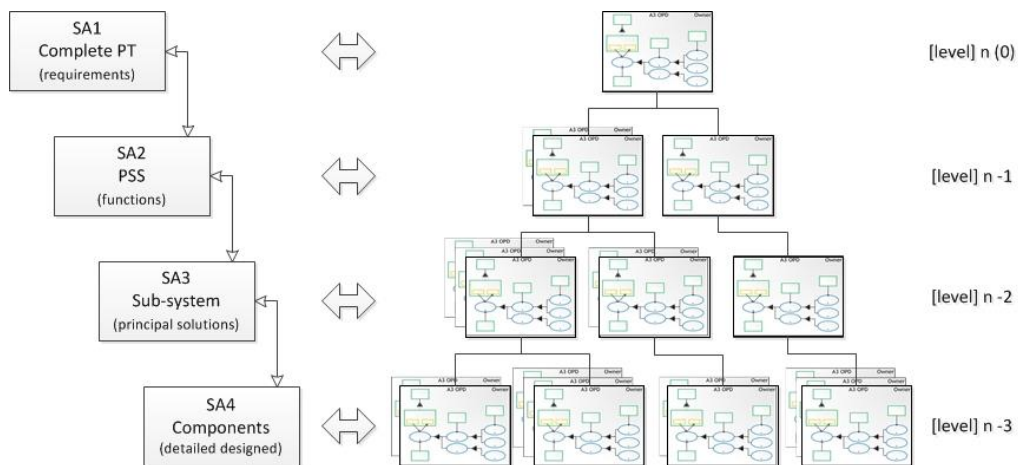
6.1.2 An A3 based hierarchical structure of object-process diagrams

In the case study, the documentation of system knowledge in OPDs follows a hierarchical structure accordingly to figure 6.3. This structure is closely connected to the systems engineering phases as well as the system responsible. The tree structure is symbolic, and may or may not represent the actual connectivity of the object and processes. Each element on a level can interface, but do not necessarily do.

The documented knowledge includes:

- System requirements, defining *why* the product is developed.
- Functions explaining *what* the solution is going to do.
- Product structure describing *how* the functions are accomplished.
- Resources expressing *who* is owner of the knowledge.
- System interfaces placing the system in a context to its surroundings.

The left side in figure 6.3 describes the communication and knowledge transfer between systems responsible. The right side describes the connections between different models at multiple system levels.



**Figure 6.3** Hierarchical structure of OPDs, following the system engineering phases in powertrain development.

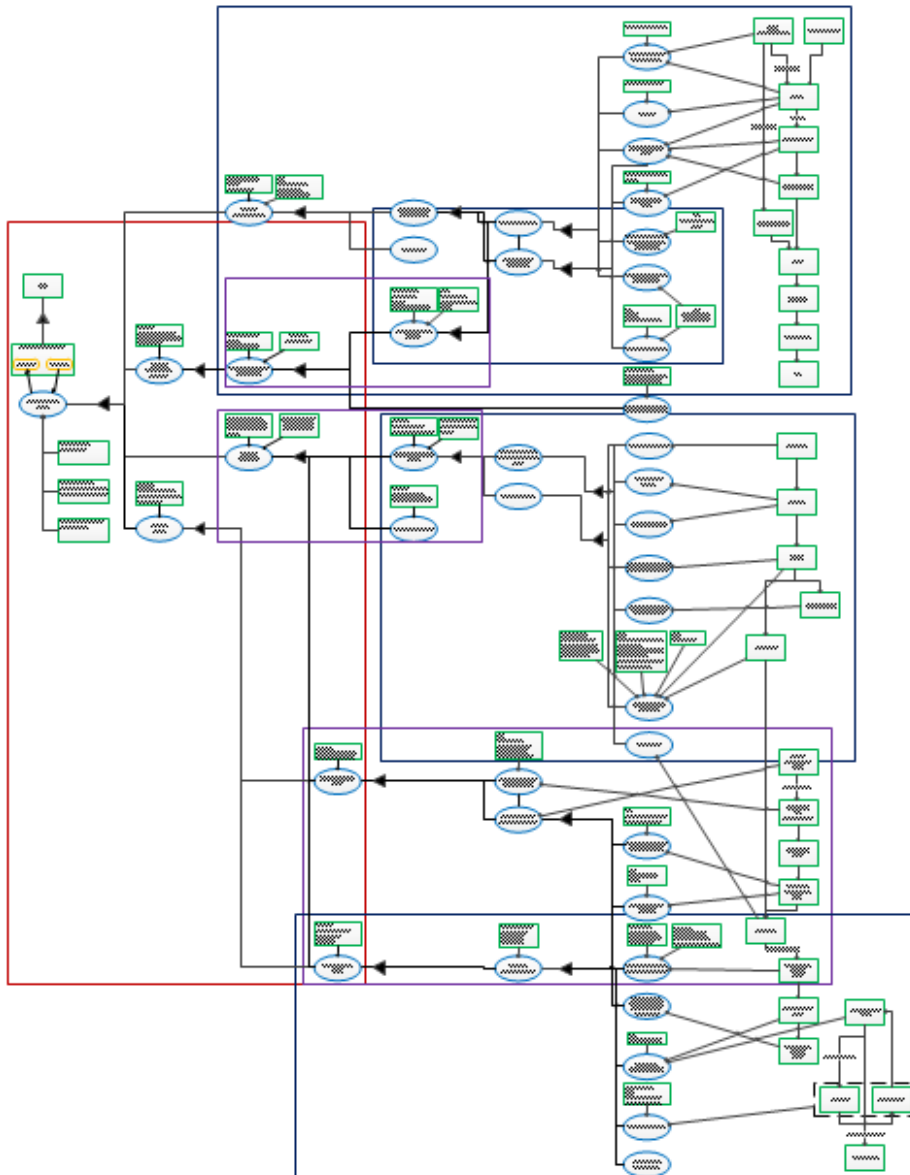
With this approach, a complete system will consist of a certain setup of A3s, reflecting the considered concepts for a certain attribute

Just to illustrate the total sum of captured knowledge in the case study, the models with different owners were merged manually to one complete system architecture. The dashed lines in figure 6.4 illustrate the different A3's and clarifies sub-systems interfaces. Similar models could exist for other attributes, for example booming noise within NVH.

In total the merged complete system architecture consists of:



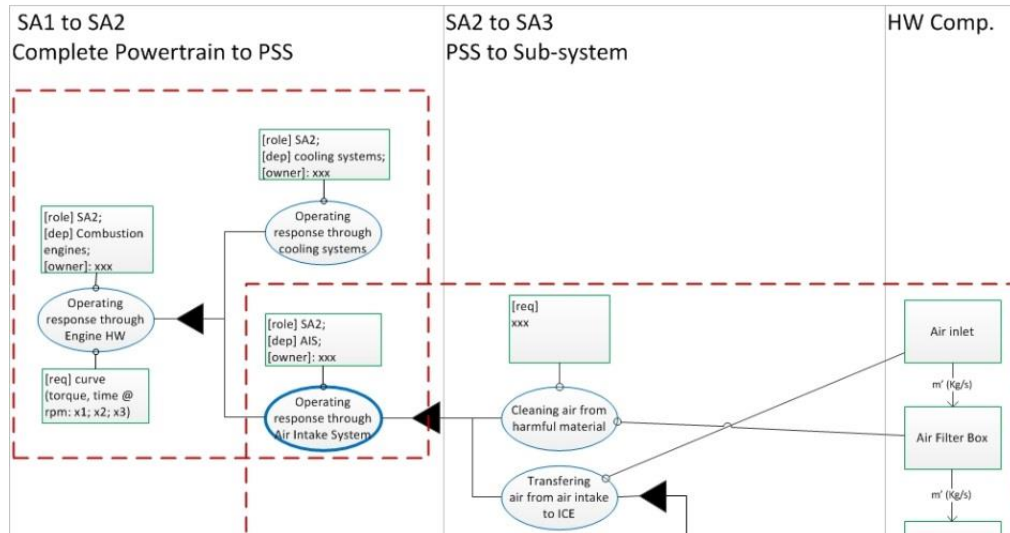
- 27 components in the product structure 41 decomposed function from the main function, transforming the attribute.
- 3 top-level requirements concerning the main function of the complete system. In the case study the top-level requirements describes load cases. Ex. take-off acceleration and road load acceleration.
- 22 derived requirements from the top-level requirements.
- 10 identified SAs delivering or receiving decomposed function and derived requirements.



**Figure 6.4** Eight object-process models merged to a complete knowledge mapped system architecture.

## 6 Knowledge mapping of a system architecture with OPM

To enable this fusion of models, duplicated elements with equivalent data content created an overlap between two systems at the same or different system level design. The function with a bold blue contour in figure 6.5 illustrates an example with output data from one model that is also input data to another model of interfacing system.



**Figure 6.5** Output from one model provides input to next system model.

Output from one model provides input to the next model through the decomposition. This overlap with duplication of the same elements, containing the equivalent information enables knowledge flow between systems, at same level or different system level design.

**Recommendation:** Send a notification to the receivers of the decomposed requirements and functions from delivering SA further up in the tree structure. Likewise, when a SA further down changes anything traced to element further up or at the same level but adjacent system. The notification is necessary so that the SA know when to loop the drafting process and continue the creation of a complete knowledge map of the system architecture.

In the case study, a function could exist in multiple systems, however the requirement or component attached to the function may differ. Example “Reducing pressure drop after compression” can be found in both air intake system and cooling system, however the required pressure drop differ. It was found that a function could be solved by multiple components and vice versa that a component could solve several functions.

Each system is responsible for some components, however several components can co- exist in several systems. Ex the component “compressor” could be found in both air intake system and turbo system. To indicate that the component belongs to an interfacing sub-system the objects was marked in a different colour.

A majority of the derived requirements were expressed explicit as  $f(x,y @ z1; z2; z3)$ , i.e. a curve from a simulation describing a multi-dimensional function, which could

facilitating tracing of trade-offs between design parameters. The requirements could also be extracted from mails, excel sheets, internal documents or simply expressed as tacit know-how.

This illustrates that it is difficult to show the complete picture of all interactions and structural relations in a complex system. Therefore the data and information in the models should be interpreted with queries.

### 6.1.3 Traceability

The object-process oriented model may provide qualitative traceability between multiple domains (described in chapter 3.4).

Traceability from vehicle attributes in the customer domain to decomposed functions and derived requirements in the functional domain. Furthermore, traceability to the product structure with hardware components in the physical domain as well as traceability to resources responsible for the development.

The models may provide qualitative traceability between systems at multiple system level design. However, a quantitative traceability, i.e. how much a change affect a system, the model will not provide. According to Almfelt [40], a quantitative traceability will require sophisticated models with integrated simulations and analysis models. Furthermore, Almfelt concludes that quantitative traceability also could be used for trade-off studies and balancing of attributes.

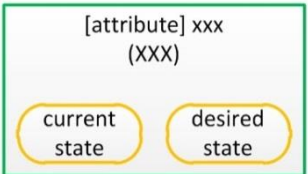
### 6.1.4 Nomenclature

To facilitate drafting of a model and ensure consistent nomenclature of the objects and processes, the following elements in table 6.1 may be used as templates. The templates ought to be drawn from a sidebar and dropped to the drawing board.

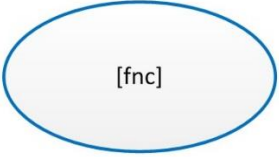
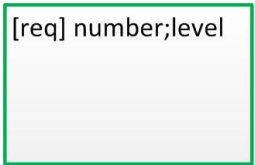
Tagged elements could enable search ability in a database storing the models and including data.

Structures for tagged nomenclature need to be aligned with existing structures within the organization, according to technical experts within Powertrain Engineering.



**Table 6.1** Templates and guidelines for nomenclature.

 <p>[attribute] xxx (XXX)</p> <p>current state      desired state</p>	<p>Tagged [attribute] ex vehicle response (driveability).</p> <p>The states expresses current state of the attribute and the desired state of the attribute. Ex: un-acceptable, acceptable. A numeric value could be included in the attribute. Ex fuel-economy xx g CO<sub>2</sub>/ km.</p>
--	--

## 6 Knowledge mapping of a system architecture with OPM

	<p>Tagged [fnc] The process element express a function necessary to meet a given requirement. Expressed as “Verb(...ing) + attribute”, more specific for each system level closer to product structure.</p> <p>Since the structured links provide traceability between the functions in different systems, numbering of functions not necessary.</p> <p>The functions with interface to the product structure are generic and expressed in a solution neutral way, I.e. they will be expressed in the same way regardless attribute described in the model, therefore requirement for different attributes may be attached to the same function. This makes them re-useable, however new functions may be included when describing another attribute.</p> <p><u>Recommendation:</u> For an implementation, a recommendation is that the supporting program suggest a named function when the author starts typing. Similar to auto-fill in excel. Another approach could be that program ask, “did you mean...” ex reducing turbo lag when author types something with maximizing turbo response.</p>
	<p>Tagged [req] number according to:</p> <ol style="list-style-type: none"> <li>1. Top-level requirement from AL for complete vehicle to SA1 at complete powertrain.</li> <li>1.1 Decomposed requirement from SA1 to SA2 at PSS area within units for powertrain engineering.</li> <li>1.1.1 Decomposed requirement from SA2 to SA3 at sub-system.</li> </ol> <p><u>Recommendation:</u> if an official explicit requirement can be retrieved elsewhere, refer with a URL to a sharepoint or to the PLM system for example.</p>

6 Knowledge mapping of a system architecture with OPM

	<p>Tagged [HW] or [SW] (capital letters, to distinguish from other objects in the model).</p> <p>During the case study, no structural links between software and hardware were mapped out. However, the method provides possibility to link software in the control system to the hardware structure.</p> <p><u>Recommendation:</u> The [HW] and [SW] ought to be managed by SA, which give the component an official name including tags for similar names referring to an equivalent component.</p> <p><u>Recommendation:</u> The product structure for the software and hardware could be linked to the engineering database (KDP) for VCG.</p>
	<p>Tagged with [owner] = name of the author to corresponding system model; [role] = the SA level (1, 2, 3 or 4); @ [dep] = name of department, number.</p> <p>The object expresses resources within the organization required for a system function.</p> <p><u>Recommendation:</u> Name and number of the departments should be linked with existing structures for business functions within the organization.</p> <p><u>Recommendation:</u> The resources may be traced to required process required to satisfy a function. Further investigation how to align the structures for activity modeling within VCG need to be investigated.</p>

### *6.1.5 Reflections from the process of capture knowledge*

Interviewee [4.2] expressed difficulties to find explicit written requirements, as they were communicated through multiple channels.

Interviewee [7] expressed that a function may be called differently even though the names aiming at an equivalent function. During the case study, equivalent functions were copied from one system and pasted in another. In addition, the nomenclature was saved in an excel file together with similar names aimed at equivalent function.

Interviewee [9] expressed that it was a useful exercise to describe their system and its functions to someone less familiar with the system. Furthermore, the interviewee expressed that the model with its OPD had similarities with boundary diagram, a method used to illustrate interface between different sub-systems.

A major difficulty during the sessions with SAs was to define the functions of the system as well as to define which requirements that are related to a specific function. Instead of defining the purpose of the system with a functional analysis the SA defined their system based on its product structure, i.e. a solution oriented approach. A solution oriented approach that may be useful when documenting a system that has been developed, however the approach contributes less to generate new ideas for principal solution.

However, the process of conducting the eight interviews and document the result in an object-process model was an iterative process that and therefore the outcome had higher quality from the last sessions.

Another area within R&D at VCG where system knowledge could be captured in an object-process model is the vehicle electrical network. A network with highly intertwined software functions with corresponding input- and output signals, according to technical expert within Powertrain Strategy and Concept.

During the project, there was a discussion whether the knowledge owners actually were willing to share their knowledge explicit. A consult hired for their expertise might not be willing to share their specific knowledge explicitly. In fact, if each individual share essential knowledge explicitly, this will mean that the owner of knowledge gets less indispensable. A conclusion is that it requires an open culture within the organization.

Another question that was brought up during discussion: “What happens if a structural link with dependency to a function is wrong? This may affect the entire system architecture in some way. However, if the system models are used as an everyday tool between the engineers, this will be brought up for discussion. The system models should be a living document that will change over time when knowledge evolves.

## 6.2 Manage and maintain captured knowledge

### 6.2.1 Support for the process

Important to clarify is that the models alone could not enable important needs such as knowledge transfer, traceability and searchability for example. Assumed is that the models are supported by a program for managing the models, such as drafting and maintenance. Furthermore, a plug-in is needed for interpreting of the data in the models.

A PLM system is needed to simplify access of models and making the knowledge available for new applications, i.e. reusable. The PLM system also facilitate version management. Ulonska [18] concluded in his study that a single place or less places for documented knowledge improves the possibility to find desired knowledge for reuse, therefore a PLM system would be appropriate.

### 6.2.2 A3 / object-process model management

In the case study, an A3 sheet with object-process diagram describes:

- One attribute, ex. vehicle response (driveability)
- One system design alternative. In this case it could be a SDA of:
  - Diesel or petrol powertrain
  - Current or next generation of powertrain
  - Configurations from entry-level design, medium performance or design for high performance.
- One knowledge owner: a SA1, SA2, SA3 or SA4.

The approach with an A3 sheet for a single system model means that each system responsible could be owner of many A3 sheets. To illustrate this approach one sub-system at a SA2 level was modelled with two different configurations. A major of the elements in the object-process model remained the same; however, expressed was the differences in component structure, additional functions and their related requirements.

To clarify the ownership of an A3 sheet, i.e. the responsibility for maintenance of data, information and knowledge in the model, a footer may be placed in the lower right corner. The footer ought to be tagged with meta-information accordingly to the example in figure 6.6. The footer inform who is owner of the captured knowledge and has the right competence for the system development. The information describes current system, SDA, attribute, version, last update as well as the source of information i.e. owner, role, department.

```
[system] complete powertrain
[SDA] MP gen 3
[attribute] vehicle respons (driveability)
[owner] name
[role] SA1 @
[dep] P/T Strategy and Concept, 97xxx
[ver] 2
[date] 2015-03-17; 2015-03-27
```

**Figure 6.6** Footer with meta-information about current object-process model.

Important to clarify is that the A3 approach does not mean that the model is drafted on a physical paper passed on to next SA. As Shook [27] describes, the A3 management means that the author or owner of knowledge and related OPD, have the explicit responsibility to describe their system area and context. The owner also has the responsibility for maintenance and updates of the content.

The decomposition of the system force the owner of the A3 to describe the reality of the system and present facts. The owner will ensure that the transferred knowledge has high technical reliability, correct data content and logical structure. If each SA creates their models it will provides many A3 sheets with explicit knowledge and contribute to the hierarchical structure of captured knowledge.

### 6.3 Interpret and reuse captured knowledge

If all the object-process oriented models with captured knowledge are merged, they will create a large system architecture. However, it can be challenging for the user to orient themselves in all data and information displayed in these merged models.

Recommendation: Captured knowledge should exist in a database and only relevant system area should appear on request.

#### 6.3.1 Interpretation

One type of query may be to interpret the trace links in the model and show dependencies in a traceability tables. According to Almfelt [40] traceability tables could be used to maintain traceability information.

Three systems in the case study were mapped manually in a traceability table. This is a DMM, displaying cross-references between two domains, with “X” to indicate trace links between items in the rows and in the columns. Due to confidentiality, the data in the tables are censored. Figure 6.7 maps links between system function and hardware components. Figure 6.8 maps links between system functions and their related requirements.

However, it is not possible to do this operation manually, when the data amount to interpret is bigger. Therefore, a plug-in is necessary to interpret the models and create the DMM.



## 6 Knowledge mapping of a system architecture with OPM

		[HW] A1	[HW] A2	[HW] A3	[HW] A4	[HW] A5	[HW] A6	[HW] A7	[HW] B1	[HW] B2	[HW] B3	[HW] C1	[HW] C2	[HW] C3	[HW] C4	[HW] C5	[HW] C6
[fnc] A1			x														
[fnc] A2		x	x	x	x	x	x	x									x
	[fnc] A2.1				x												
	[fnc] A2.2			x	x												
	[fnc] A2.3					x											
	[fnc] A2.4						x	x									
[fnc] B1						x			x	x	x						x
	[fnc] B1.1								x								
	[fnc] B1.2									x	x						
	[fnc] B1.3																
[fnc] C1												x	x	x	x	x	x
	[fnc] C1.1											x					
	[fnc] C1.2												x				
	[fnc] C1.3												x				
[fnc] C2														x			x
	[fnc] C1.4; C2.1													x			
	[fnc] C1.5														x		
	[fnc] C1.6; C2.2															x	
	[fnc] C1.7																x

**Figure 6.7** Traceability table between decomposed functions and HW components.

		[req] A1	[req] A2.1	[req] A2.2	[req] B1	[req] A2.3; B1	[req] B1.1	[req] B1.2	[req] C1	[req] C1.6.1	[req] C1.6.2	[req] C1.6.3
[fnc] A1		x										
[fnc] A2			x	x								
	[fnc] A2.1		x									
	[fnc] A2.2			x								
	[fnc] A2.3					x						
	[fnc] A2.4											
[fnc] B1					x	x						
	[fnc] B1.1						x					
	[fnc] B1.2							x				
	[fnc] B1.3											
[fnc] C1									x			
	[fnc] C1.1											
	[fnc] C1.2											
	[fnc] C1.3											
[fnc] C2										x	x	x
	[fnc] C1.4											
	[fnc] C1.5											
	[fnc] C1.6								x	x	x	
	[fnc] C1.7											

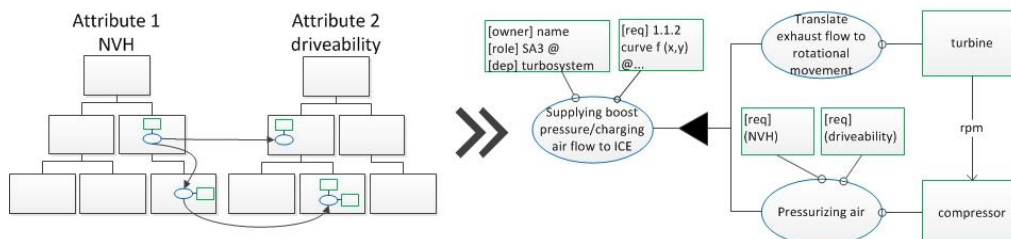
**Figure 6.8** Traceability table between decomposed functions and requirements.

### 6.3.2 Searchability

One query may be to search in the database storing the information. With interpretation of trace-links this could extract dependencies for only requested explicit system knowledge, based on multiple attributes.

Search for:

[attribute] booming noise (nvh) + [attribute] vehicle response (driveability)  
+ [HW] compressor + [SDA] HP Turbo + [role] SA3; [level] n -2



**Figure 6.9** Traceability and searchability may generate requested knowledge.

This means that an object-process diagram could be generated on demand, based on the searched tags, displaying only what is relevant documented knowledge.

Every system operates as an element of a larger system and is itself composed of smaller systems [43]. The searchability implies to choose how much should be displayed in the model. Traceability to the extent of only searched element, one system, surrounding system at the same level or further up/down in system structure.

According to system architects at the unit Strategy & Vehicle Concepts within R&D, VCG (50), the queries may be formulated as:

- Which object/processes of type [tag] *depends on* type [tag]?
- Choose level: ex. first or second order of dependency.

Example (see figure 6.9), if only interested in what the compressor in the turbo system is affected by, or affect in the immediate surrounding it is not necessary to display the complete system architecture. That will display too much information and complicate for the user to orient themselves in the system landscape.

## 7 Conclusions

*This chapter concludes the research and answering the research questions.*

In this research, a process for capture and reuse system knowledge at Powertrain Engineering within R&D at Volvo Car Group has been investigated. This with the use of object-process methodology as a tool for model based systems engineering.

*RQ1: What are the needs for the process of system knowledge capture and reuse?*

Need findings from the target group within VCG, concluded that important for the process is methods and tools supporting communication with stakeholders as well as shared understanding of a complex system. Furthermore is both traceability and searchability necessary for managing the system knowledge.

*RQ2: How can system knowledge be captured and reused by OPM?*

### Capture knowledge

The case study of mapping system knowledge with OPM concluded that conversion of tacit to explicit system engineering knowledge could be enabled by object-process oriented modeling.

The documentation of captured system knowledge should follow a hierarchical tree structure that is closely connected to the development phases as well as the system responsible.

Value offered with the object-process oriented models of a system architecture is qualitative traceability between:

- System requirements, defining *why* the solution is developed.
- Decomposed functions explaining *what* the solution is going to do.
- Product structure describing *how* the functions are accomplished.
- Resources expressing *who* is owner of the knowledge.
- System interfaces placing the system in a context to its surroundings.

A generalized structure for the decomposition of a system architecture has been presented, following VCG developing organisation within R&D.

## 7 Conclusions

---

Drafting of the model could be facilitated with “what” and “how” questions [46], as well as the additional “who” and “why” questions for decomposition of the system.

- What should the system affect?
- What transformation should the system cause?
- How does the system behave?
- How is the system structured?
- Why does a function exist/
  - Which derived requirements are related to the decomposed functions?
- Who is the resource delivering/receiving the requirements?

Templates with guidelines for defining namespace and managing an equal nomenclature of the object-process element has been discussed. Suggested is that the nomenclature is aligned with existing business structures, however it needs further investigation for a robust implementation.

### Manage and maintain captured knowledge

To transfer the system knowledge, models are stored suitably in existing PLM system, which will facilitate access and version management.

Proposed is that the model is maintained by the system responsible with A3/OPM management, i.e. the owner of knowledge. This will ensure correct content.

### Interpret and reuse captured knowledge

To support reuse of captured system knowledge and searchability, suggested is that a plug-in tool should interpret data and information in the models based on queries.

Proposed is that the interpreted system knowledge is presented with:

- A traceability table, a DMM displaying dependencies between two domains.
- An object-process model generated on demand, based on searched tags and dependency order.

The models could be used as an everyday tool for the engineers to orient in the architecture while discussing the concerned systems.

When documented explicit knowledge is transferred between SAs, the tool may provide a holistic view of the overall system architecture.

It could also support shared understanding and facilitate consensus about a complex problem when communicating the system with stakeholders.

Figure 7.1 concludes the process of capture, transfer and re-use system knowledge. Figure 7.2 concludes the process with an object process oriented model.

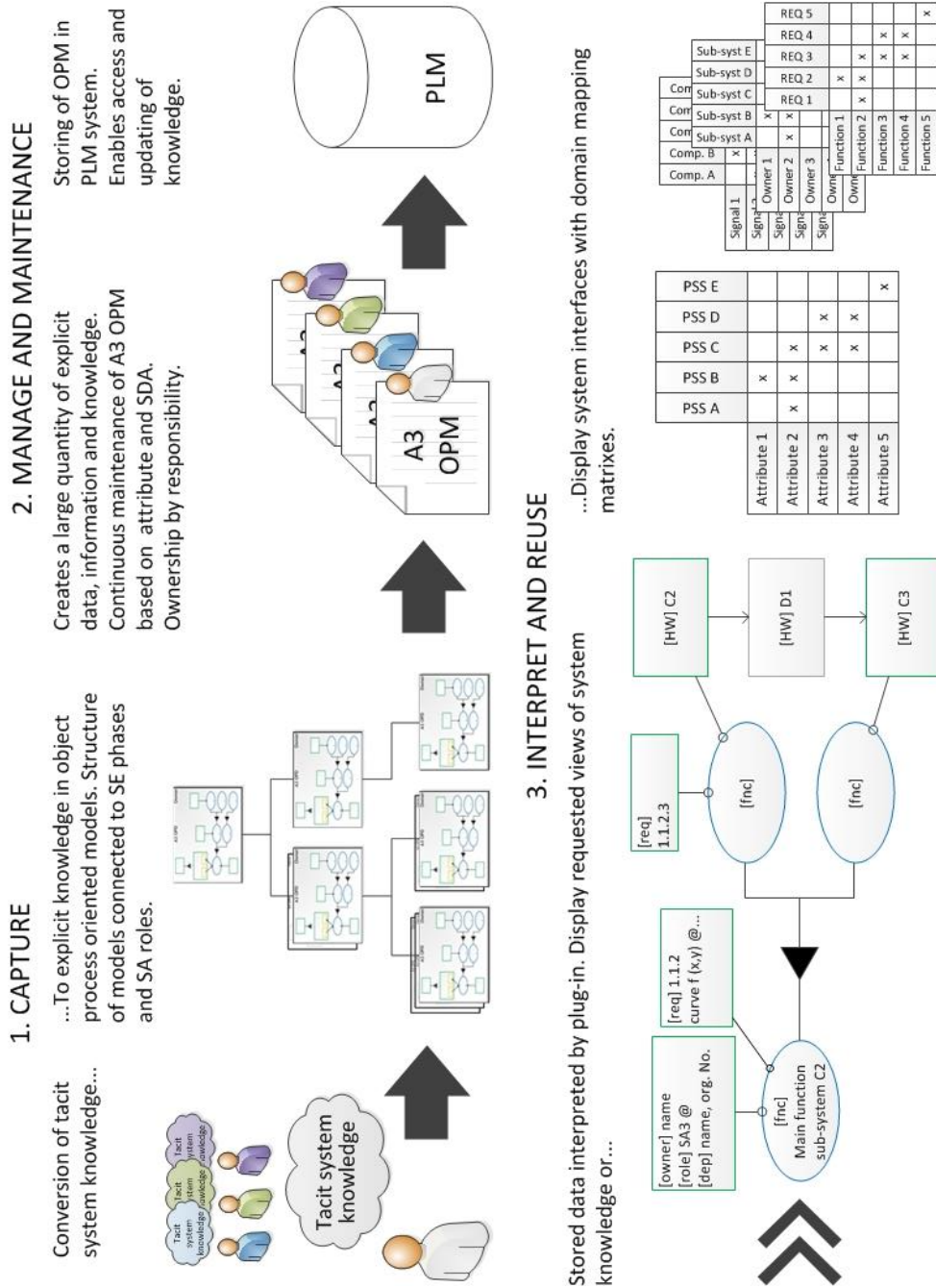
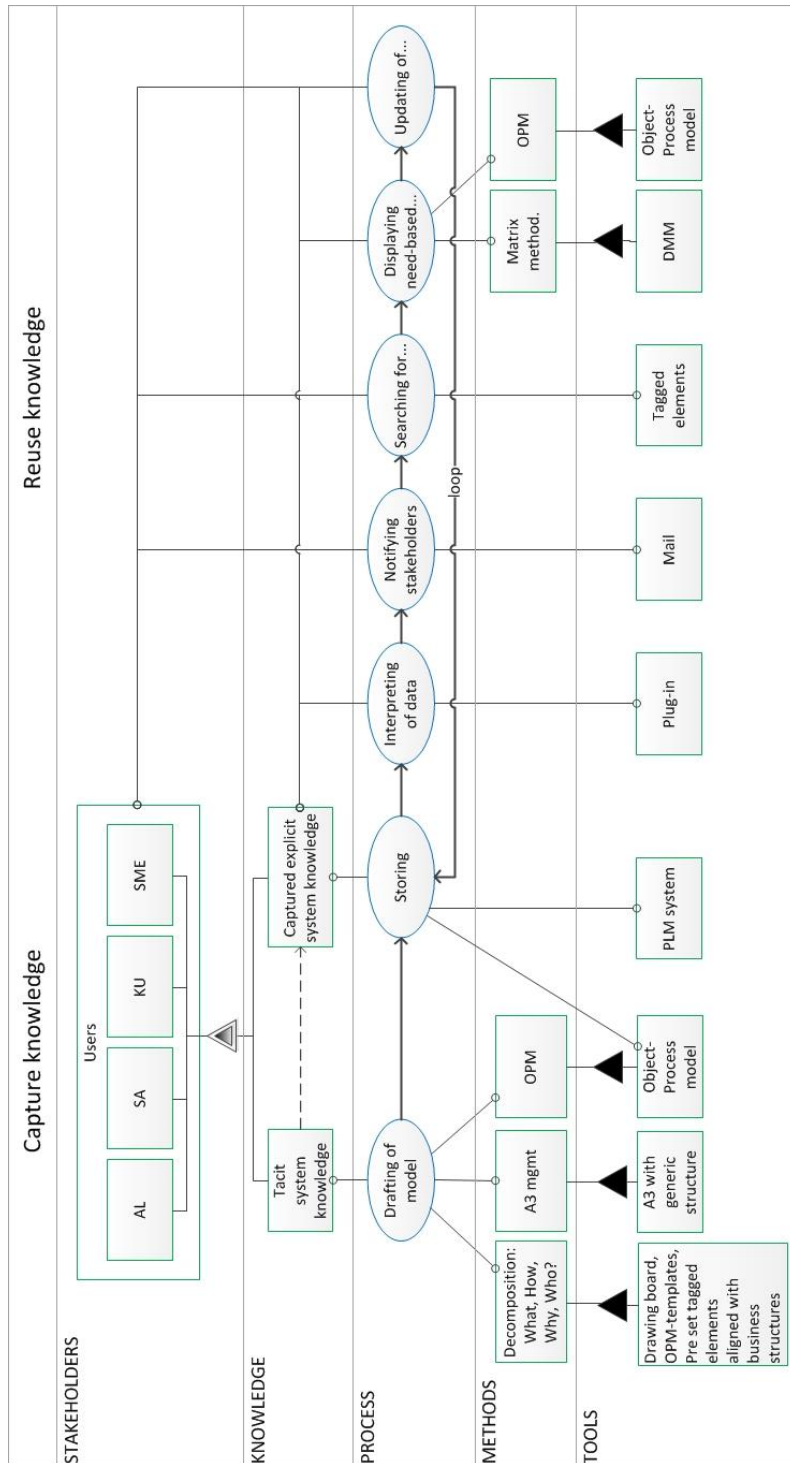


Figure 7.1 Capture, transfer and reuse system knowledge.



**Figure 7.2** Process of capture, manage, maintain, interpret and reuse system knowledge with an object-process model.

## 8 References

- [1] Volvo Cars. (2015). This is Volvo Cars. [Electronic].  
<http://www.volvocars.com/intl/about/our-company/this-is-volvo-cars>.
- [2] Volvo Car Group. (2014). Corporate Presentation. [Electronic]. Intranet.
- [3] Volvo Car Group. (2014). Research & Deveopment. [Electronic] Intranet.
- [4] Volvo Car Group. (2015). Transformation 202020. [Electronic] Intranet.
- [5] Volvo Car Group. (2014). Town Hall presentation 2014. [Electronic] Intranet.
- [6] Dori, D. (2005). Mapping Knowledge about Product Lifecycle Engineering for Ontology Construction. *CIRP Annals – Manufacturing Technology*, 54 (1), pp.117–122
- [7] Dori, D. (2002). *Object-Process Methodology: A Holistic Systems Paradigm*. Springer-Verlag. Berlin-Heidelberg. ISBN 978-3-642-62989-1
- [8] Törmnen, M; Johansson, K.; Möller, P.; Krizmanic; M. (2013). Investigation on the use of logical models, Volvo Car Corporation [Electronic] Intranet.
- [9] Törmänen, M. Technical Expert MDO & P/T Attributes Balancing, Powertrain Strategy & Concept, Volvo Car Group, personal conversation (2015)
- [10] Denzin, N. K. and Lincoln, Y. S. (2000). *The SAGE Handbook of Qualitative Research*. Sage Publications. Thousands Oaks. ISBN: 0761915125.
- [11] Willis, J. (2007). *Qualitative Research Methods in Education and Educational Technology*. Information Age Publishing. ISBN: 9781930608542
- [12] Ridley, D. (2012). *The Literature Review: A Step-by-Step Guide for Students*. SAGE Publications. ISBN: 9781446281093.
- [13] Hove, S. E. and Anda, B. (2005) Experiences from Conducting Semi-structured Interviews in Empirical Software Engineering Research. *Proceedings of the 11th IEEE International Software Metrics Symposium*, 19-22 Sep, Como, pp. 10
- [14] Fontana, A and Frey, J.H. (1994) Interviewing - the Art of Science. In Denzin, N. K. and Lincoln, Y. S. (eds.). *The SAGE Handbook of Qualitative Research*. Sage Publications. Thousands Oaks. ISBN: 0761915125
- [15] Ulrich, K. and Eppinger, S. (2011) *Product Design and development*: McGraw-Hill Higher Education, New York. ISBN: 9780071086950

## 8 References

---

- [16] Eckhardt, C-C. (2013) Design Methodology Compilation. Lund University School of Industrial Design. Lund
- [17] Yin, R. K. (2013) Case Study Research. SAGE Publications, Thousands Oaks. ISBN: 9781452242569.
- [18] Ulonska, S. (2014). A knowledge based approach for integration of system design methodology and documentation in advanced multi-disciplinary NPD projects. Doctoral Thesis. Department of Engineering Design and Materials. Norwegian University of Science and Technology, Trondheim. ISBN 978-82-326-0662-7.
- [19] Schubert, P.; Lincke, D-M. and Schmid, B. (1998) A Global Knowledge Medium a Virtual Community: The Net Academy concept. AMCIS Proceedings. Baltimore, Paper 207.
- [20] Rowley, J. (2007) The wisdom hierarchy: representations of the DIKW hierarchy. Journal of Information Science. 33 (2), pp. 163-180.
- [21] Ackoff, R.L. (1989) From data to wisdom. Journal of Applied Systems Analysis. 16, pp. 3-9.
- [22] Nonaka, I. and Teece, D. J. (2001) Managing Industrial Knowledge: Creation, Transfer and Utilization. Sage Publications, Thousands Oaks. ISBN 076195498 8.
- [23] Nonaka, I.; Toyama, R. and Konno, N. (2000) SECI, Ba and Leadership: a Unified Model of Dynamic Knowledge Creation. Long Range Planner, 33 (1).
- [24] Nonaka, I. and Takeuchi, H. (1995) The knowledge creating company. Oxford University Press, New York. ISBN 9780195092691
- [25] Kennedy, M.; Harmon, K. and Minnock, E. (2008) Ready, Set, Dominate: Implement Toyota's Set-Based Learning for Developing Products and Nobody Can Catch You. Oaklea Press, Richmond. ISBN 1892538407.
- [26] Grant, R. M. (1996) Towards a knowledge based theory of the firm. Strategic management journal. 17, pp. 109
- [27] Shook, J. (2009) Lean Management med hjälp av A3 analyser. Liber, Malmö. ISBN 978-91-47-09449-3.
- [28] Stevens, R.; Brook, P.; Jackson, K. and Arnold, S. (1998) Systems engineering - coping with complexity. Prentice Hall Europe. Essex ISBN: 978-0-13-095085-7.
- [29] Haskins, C. and Forsberg, K. (2011) Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities. INCOSE.
- [30] Gianni, D.; D'Ambrogio, A. and Tolk, A. (2014) Modeling and Simulation-Based Systems Engineering Handbook. CRC Press. ISBN 978-1-4665-7145-7.



- 
- [31] Bruun, H. P. L. (2014). PLM based support to architecture based development. Doctoral Thesis. Mechanical Engineering. Technical University of Denmark, Copenhagen . ISBN 978-87-7475-397-1.
- [32] Börjesson, F. (2014). Product platform design. Doctoral Thesis. Machine Design. KTH Royal Institute of Technology Stockholm. ISBN: 978-91-7595-113-3.
- [33] Malmqvist, J. (2002) A classification of matrix based methods for product modeling.: Proceedings of the 7<sup>th</sup> International Design Conference, Dubrovnik, Croatia, May 14-17, (214) pp. 203-210
- [34] Suh, N. P. (1990) The Principles of Design. Oxford University Press, New York. ISBN: 9780195043457.
- [35] Sustainable Innovative Solutions Limited (2008) Axiomatic design. Design for innovation. [Electronic] <https://designforinnovation.wordpress.com/>.
- [36] Hommes, Q. V. E. (2010) ESD.33 Systems Engineering, lecture 6, requirements driven design. MIT OpenCourseWare. [Electronic] <http://ocw.mit.edu/courses/engineering-systems-division/esd-33-systems-engineering-summer-2010/>.
- [37] Kapurch, S. J. (2007) NASA Systems Engineering Handbook. NASA. Washington D.C. NASA/SP-2007-6105 Rev 1.
- [38] Chestnut, H. (1965) Systems Engineering Tools. Wiley, New York. ISBN: 0471154466.
- [39] Hull, E.; Jackson, K. and Dick, J. (2002) Requirements Engineering. Springer-Verlag, London. ISBN: 978-1-85223-577-9.
- [40] Almfelt, L. (2005) Requirement-driven product development - methods and tools reflecting industrial needs. Chalmers University of Technology Gothenburg. ISSN 0346-718-X.
- [41] Leonard, J. (2001) System Engineering Fundamentals. Defense acquisition university press, Fort Belvoir, Virginia ISBN: 1484129555.
- [42] Ulrich, K. (1995) The role of product architecture in the manufacturing firm. Research Policy. 24(3), pp. 419-440
- [43] Crawley, E. (2007) MIT OpenCourseWare. MIT Course Number ESD.34 Lecture Notes. [Electronic]. <http://ocw.mit.edu/courses/engineering-systems-division/esd-34-system-architecture-january-iap-2007/>.
- [44] Weck, O. (2009) MIT OpenCourseWare. 16.842 Fundamentals of Systems Engineering. [Electronic] <http://ocw.mit.edu/courses/aeronautics-and-astronautics/16-842-fundamentals-of-systems-engineering-fall-2009/>.
- [45] Soderborg, N. R. (2002) Representing systems through Object Process Methodology and Axiomatic design. Master Thesis. System Design and Management. Massachusetts Institute of Technology.

## 8 References

---

- [46] Soderborg, N. R; Crawley, E. and Dori, D. (2002) System definition for axiomatic design aided by Object Process Methodology. Proceedings of ICAD2002. Second International Conference on Axiomatic Design Cambridge, MA – June 10 &11.
- [47] Eriksson, S. (2007) Powertrain Systems Engineering at Volvo Car Corporation. [Electronic] Intranet
- [48] Volvo Car Group. (2015) Business Management System. [Electronic] Intranet.
- [49] Maxwell, M. and Sjögren R. SA1 Powertrain Strategy & Concept, Volvo Car Group. Interview. (2015)
- [50] Nilsson, R. and Batejat, F. System Architects, Strategy and Vehicle Concepts, Volvo Car Group. Personal conversation. (2015)

## Appendix A: Interviews

No.	Role		Occ.	Date
1.1	SA1 (phase A) Strategy and Concept, Complete powertrain	Un-structured	3	15-02-18; 15-03-04; 15-03-16
1.2	SA1 (phase B) Strategy and Concept, Complete powertrain	Un-structured	3	15-02-18; 15-03-04; 15-03-16
2	SA2 Air Intake System, DLI	Semi-structured	1	15-03-16
3	SA2 Complete drivetrain, KD	Semi-structured	1	15-03-20
4.1	SA2 Propulsive drive and torque, TU	Semi-structured	1	15-03-23
4.2	SA2 Automatic Transmission Design, TU	Semi-structured	1	15-03-23
5	SA3 Ratio Management, KD	Semi-structured	1	15-03-23
6	SA2 Cooling System, DLI	Semi-structured	1	15-03-24
7	SA3 Calibrator, former Attribute Leader Driveability, KD	Semi-structured	1	15-03-27
8.1	SA2 Diesel Engine, MU	Semi-structured	1	15-04-08
8.2	SA2 Petrol Engine, MU	Semi-structured	1	15-04-08
9	SA3 Turbo system, MU	Semi-structured	1	15-04-15



## **Appendix B: Questions addressed to SA1 for need findings**

*Question addressed to the target group with means to find needs:*

What do you want to communicate with the model?

With whom do you communicate?

What do you want to document in the model?

When do you want to use the model?

How do you want to use the model?

What is important when maintaining the model?



## Appendix C: Interview guide used in case study

### Interview with SA1:

- What is the overall attribute for complete vehicle from the Attribute Leader?
  - Ex. Vehicle response (driveability).
- Who or What perceives/experiences the value of this attribute?
  - Example: the driver.
- How are the states of the attributes expressed?
  - Before and after desired state has been fulfilled.
  - Ex acceptable or un-acceptable. Premium or non-premium.
- What transformation of attribute should occur to reach desired state?
  - Ex operating engine response within acceptable time frame.
- What are the main requirements for the transformation of the attribute? What are the use cases when the transformation is needed?
  - Ex different road load accelerations.
- How are these requirements decomposed to development units and/or PSS areas?
- How are these requirements expressed?
  - Ex as a curve.
- Is there any trade-offs regarding these requirements?
  - Ex time to torque vs CO<sub>2</sub> levels.
- What function is the SA2 responsible of?
  - Ex operating engine hardware for reducing time to torque.
- What are the major system design alternatives for this PSS area?
  - Ex diesel or petrol engine.
  - Entry level, medium performance or high level performance.
  - Current or next generation.

### **Interview with SA2:**

(Sketch a schematic illustration of the system)

- Regarding the overall attribute (ex vehicle response → engine response) and current PSS area, which sub-systems provides a direct or indirect impact on the attribute?
- What are the main purposes of these systems? What is the main function of the sub-system/what does the system perform?
  - Ex provide charged air and boost pressure to engine.
- What are the system design alternatives for this sub-system?
- Which function has the biggest impact on the attribute?
- Does other SA2s deliver requirements for the current sub-system?
- How are these requirements expressed?
- Is there any trade-offs regarding these requirements?
- Who (name/SA2 role/department) is responsible for delivering these requirements?
- Who (name/SA3/department) is receiving these requirements?
- If the subsystem is complex, is there any function owners responsible for development of a specific function?
- Which are the SDAs'?

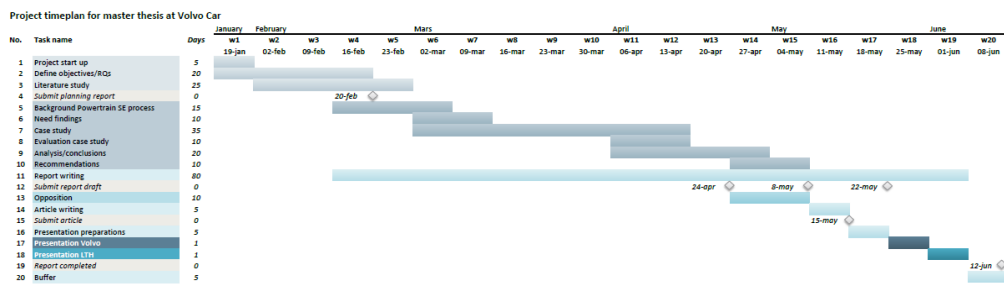
### **Interview with SA3:**

(Sketch a schematic illustration of the system)

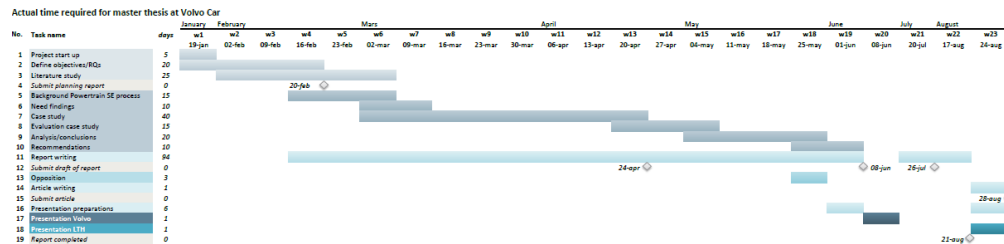
- What is the main purpose/function of the systems?
- Regarding attribute (vehicle response → engine response → turbo response) How does the system need to be operated?
  - Ex. Reducing turbo lag, reducing inertia of rotating parts.
- Which function has the biggest impact on the attribute?
- How is the product structure decomposed?
- What is the function/task/goal of the parts?
- What is the excitation to the part?
- How does the part transfer energy?
- What is the physical interface between the parts?
  - Ex mass flow, pressure, force, etc.
- Is there any derived requirements concerning these parts and their functions?
- How are these requirements expressed?
- Are there any trade-offs?
  - Ex inertia vs structure borne vibration and booming noise.
- Which are the SDAs?



## Appendix D: Gantt chart for thesis project



Gantt chart for estimated time (20 weeks) required for master thesis project.



Gantt chart for actual time (23 weeks) required for maser thesis project.

Analysis and conclusions required more time than expected. A structured approach, defined at an early stage for this part of the project, would have been useful.