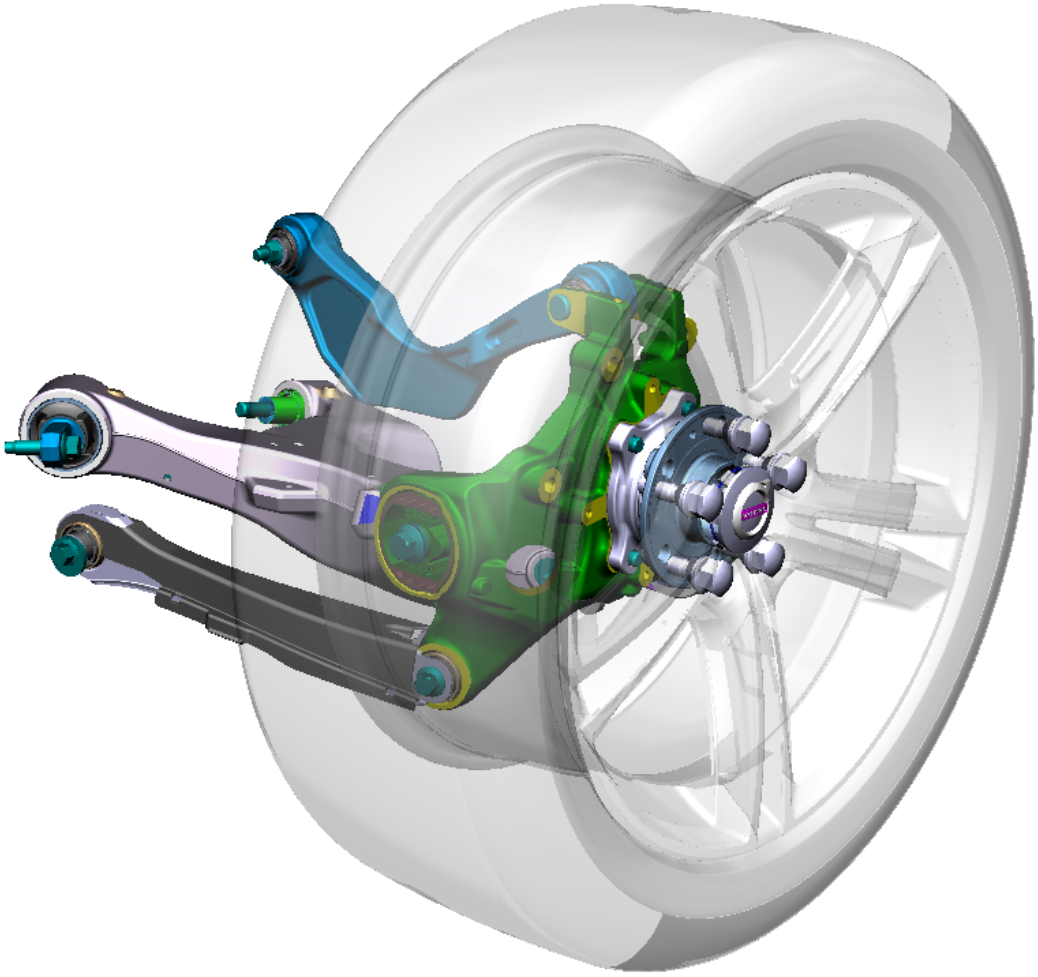
CHALMERS
UNIVERSITY OF TECHNOLOGY



Optimization of Wheel Suspension Packaging

Methodology Development, Data Transfer from ADAMS/Car to CATIA V5

Master thesis in Product Development

KARL HANSEN ANDREASSON
MATTIAS LINDER

Department of Product & Production Development
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2016

MASTER'S THESIS 2016

Optimization of Wheel Suspension Packaging

Methodology Development, Data transfer from ADAMS/Car to CATIA V5

KARL HANSEN ANDREASSON
MATTIAS LINDER

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Product & Production Development
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2016

Optimization of Wheel Suspension Packaging
Methodology Development, Data transfer from ADAMS/Car to CATIA V5
KARL HANSEN ANDREASSON
MATTIAS LINDER

Supervisor: Daniel Molin, Volvo Cars
Supervisor: Magnus Bengtsson, Department of Product & Production Development
Examiner: Magnus Bengtsson, Department of Product & Production Development

Master's Thesis 2016
Department of Product & Production Development
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: SPA Rear-wheel suspension.

Typeset in LaTeX
Printed by Reproservice
Gothenburg, Sweden 2016

Optimization of Wheel Suspension Packaging
Methodology Development, Data transfer from ADAMS/Car to CATIA V5
KARL HANSEN ANDREASSON
MATTIAS LINDER
Department of Product & Production Development
Chalmers University of Technology

Abstract

Finding an optimized and balanced concept of the wheel suspension, and its surrounding components, is of great importance for future competitiveness. In order to do this, requirements need to be met in the early phases of development and design loops need to be shortened. This is a pre-requisite for enabling cost, weight and lead time reductions while also allowing for more design loops to be carried out, thus reducing the risk for late changes.

A complex system, such as a wheel suspension, requires a methodology which enables CAE driven development on several levels, where a natural part is design optimization and a tight coupling between design and analysis (CAD & CAE).

This master thesis aims at establishing the tight coupling through development of an application which can transfer motions from a multi-body dynamics simulations, in ADAMS/Car, to a kinematic model in CATIA V5.

The work has covered the entire development process, from pre-study and requirement specification to concept development, detail design and testing & verification. Throughout the development effort there has been an ongoing dialogue with future users, as to understand their needs and develop a user-centered product.

The end result is an application which increases precision of packaging analysis while also reducing the lead times of packing critical components of a wheel suspension, thus shortening design loops. The application is ready to be incorporated into Volvo Cars working methodology.

Keywords: Optimization, CATIA V5, ADAMS/Car, kinematics, wheel suspension, data transfer.

About the Authors

Karl Hansen Andreasson

I received my BSc in Mechanical Engineering at University West in Trollhättan, thereafter I pursued my MSc in Product Development at Chalmers University of Technology in Gothenburg. I would like to thank my grandfather Paul for inspiring me to pursue engineering at the university. I also want to thank my family for their love and support throughout my education, specifically my mother Cindy and my father Janne who have been my biggest supporters.

Mattias Linder

I received my BSc at the program Automation and Mechatronics at Chalmers Univeristy of Technology. Afterwards I continued with the MSc program Product Development at Chalmers. I would like to thank my dad Thord, for inspired me to study at Chalmers and become an engineer. I would like to thank my mother Gunnel who has supported me through all my educations. At last, I would like to thank my sisters Jennifer and Josefin for their love and support.

Acknowledgements

First of all we would like to thank our supervisor at Volvo Cars, Daniel Molin. Who has supported us on a daily basis throughout the entire project. We would also like to thank our supervisor at Chalmers University of Technology, Magnus Bengtsson for his guidance. Further we want to thank David Fredriksson and Albin Johnsson, at the Vehicle Dynamics Department, for their help within ADAMS/Car. We would also like to thank Fredrik Sjögren at MSC Software for his collaboration. Finally we would like to thank Harald Hasselblad of Volvo Cars Optimization Culture Arena for giving us the opportunity to take part in its Master Thesis Collaborations.

Karl Hansen Andreasson, Gothenburg, June 2016

Mattias Linder, Gothenburg, June 2016

Nomenclature

$\hat{x}$	The unit vector for the x-direction
$\hat{y}$	The unit vector for the y-direction
$\hat{z}$	The unite vector for the z-direction
ψ	Represents the rotation around the third axis in a given rotation sequence
θ	Represents the rotation around the second axis in a given rotation sequence
φ	Represents the rotation around the first axis in a given rotation sequence
API	Application Programming Interface
CAD	Computer Aided Design
CAE	Computer Aided Engineering
COG	Center Of Gravity
CS	Coordinate System
DoE	Design of Experiments
$Jounce$	Suspension Compression
KNU	Knuckle
LCA	Lower Control Arm
OEM	Original Equipment Manufacturer
$Rebound$	Suspension Extension
UCA	Upper Control Arm

Contents

List of Figures

List of Tables

1
Introduction

This chapter presents the scope of the work and the cluster of master theses in which this thesis has taken part, as well as some company history about Volvo Cars.

1.1 Background

Volvo Cars have set the goal of a time-to-market for new models to 20 months in the year 2020 [1]. This is a reduction of 12 months compared to the current time-to-market of 32 months (2016). In order to achieve this great investments are being made into increasing the efficiency and effectiveness of the company's current working methods, as well as establishing new methods were necessary. An important part in this work is to enable CAE integration early in the development process, which reduces the need for costly and time consuming physical testing in later design stages. In order to integrate CAE early on the importance of data and information transfer between analysis and design (in Volvo cars' case ADAMS/Car and CATIA) becomes a central issue.

1.2 Purpose

The purpose of this master thesis is to optimize the working methodology for generating design spaces of critical components in the wheel suspension. This can be broken down into two main objectives both of which concern data transfer from CAE (ADAMS/Car) to CAD (CATIA) software. The simulation results from ADAMS/Car produce more realistic results as flexible bodies are used and bushing stiffness is accounted for. Due to limitations in CATIA V5's Kinematics Workbench these two factors must be accounted for by tolerances and generic rules.

The first objective is to reduce the lead time of producing a design space for an arbitrary component. This will be done by automating the data transfer from ADAMS/Car to CATIA V5, resulting in shorter lead times (in the range of 50 effective working hours for a critical component [2]).

The second objective is to increase the accuracy of the packaging analysis in CATIA V5. This will be done by using the transferred data as input, instead of mimicking the movement within CATIA V5's Kinematics Workbench, which is the case today. By doing so the tolerances and generic rules can be replaced by actual motions.

1.3 Aim

In order to fulfill the objectives stated in Section 1.2 an interface which allows motions from drive-case simulations in ADAMS/Car to be transferred into CATIA and used to control the wheel suspension kinematics. This interface must:

- Read data from ADAMS/Car, either from a standard file or through requesting certain information.
- Enable mapping of CATIA instances to motions extracted from ADAMS/Car.
- Automatically generate a replay file in CATIA V5.
- Incorporate post-simulation analysis into the solution.
- Increase the capacity of data handling, compared to today's working process.
- Minimize information loss during data transfer.
- Be intuitive to handle and not require extensive training.
- Allow for further improvements to be added in the future.
- Not deviate significantly from the current working procedure and today's role distributions.

1.4 Limitations

- Tests during the project have been limited to the rear wheel suspension of Volvo Car's SPA platform, set up in the S90/V90 configuration.
- The tests carried out have been limited to ADAMS/Car sub-system models of the wheel suspension, with an emphasis on kinematics and compliance. Full vehicle models are not included.
- This project will not include hardpoint optimization as it is limited to developing a methodology which will help to enable this.
- The project is limited to working with software which is currently available at Volvo Cars Wheel Suspension department.
- The project is limited to 20 weeks of full time work by two Master students.
- The scope of this project has been limited to transfer motions of rigid bodies from ADAMS/Car, however preparations for including body flex have been carried out.

1.5 Research Questions

- How can packaging of the wheel suspension be optimized in a fast and flexible way with respect to system dynamics, forces and moments?
- How can data be transferred from ADAMS/Car to CATIA V5 and which is the most efficient way, with respect to user friendliness and efficiency?
- How can knowledge from other areas/applications be automatically integrated into CATIA V5?

1.6 Related Master Theses

This master thesis is part of a cluster consisting of three master theses within the area of design optimization. The three theses focus on different stages of the wheel suspension development process. This master thesis, ”Optimization of Wheel Suspension Packaging”, focuses on methodology development within data transfer from CAE to CAD software. The second master thesis, “Balancing of Wheel Suspension Packaging, Performance and Weight” [3], focuses on methodology development concerning the balancing of design spaces within a system. The third master thesis, "Structural Topology and Shape Optimization" [4], concerns methodology development regarding structural optimization of an individual component. The intention is to, sometime in the future, implement the methodologies developed in the three theses and thus create a complete working process from hardpoint placement down to topology optimization on component level.

1.7 Optimization Culture Arena

The three theses in the cluster have been part of a larger “Optimization Culture Arena” at Volvo Cars. The aim of the “Optimization Culture Arena” is to share knowledge, support implementation and develop methods for CAE driven optimization across Volvo Cars entire organization. The scope is to involve all areas and work towards fully cross-functional optimization. Within this arena meetings are held where master theses which involve CAE driven optimization meet and share knowledge as the projects progress.[5]

1.8 Volvo Cars

Volvo, Latin for: “I roll”, was founded in 1927 as a subsidiary to the Swedish ball bearing manufacturer SKF. The two founders Assar Gabrielsson and Gustaf Larson planned to manufacture a Swedish car which could withstand the harsh winter and bad road conditions of the north. The emphasis was therefore, from the very beginning, set on safety and durability. [6]

> *"Cars are driven by people. The guiding principle behind everything we make at Volvo, therefore, is and must remain, safety"*
>
> - Assar Gabrielsson and Gustaf Larson, *1927*

Volvo soon began to broaden its production to trucks and thereafter further by acquiring production within engines and components for air crafts, as well as marine engines. After the end of the Second World War Volvo experienced rapid growth partly due to the successful introductions of the Volvo PV444, PV544 and the Amazon, which was regarded as one of the safest cars of the time. In 1954 Volvo had 4500 employees and a turnover of 483 Million SEK [7]. By the beginning of the 1970s Volvo had 40 000 employees and a turnover of 6 Billion SEK [7]. Volvo experienced some tougher years during the oil crisis of the 1970s with a low number of model launches and focus set on acquisitions within completely new industries, e.g. the pharmaceutical industry. This was part of Volvo’s plan to diversify the company, as the automotive industry became more and more competitive.

However, during the 1980s Volvo’s success within the automotive industry increased and the company manufactured record breaking volumes twice during the decade. The first time in 1983 and the then again in 1987, when the company manufactured 372 400 cars and 423 800 cars, respectively. It was also during the 1980s that Volvo launched its famous 740 model. [7]

By the beginning of the 1990s Volvo sold its ownership in the previously mentioned acquisitions which were not related the Volvo’s core business, the company kept its business within trucks, buses, airspace engines, marine engines and cars. During the 1990s the 740 was replaced by the 940, both cars were known for their reliability and robustness. These models are still seen on the road today, especially in Sweden. During 1993 Volvo came close to merge with the French car manufacturer Renault, the merger was waved off days before it was to take place due to large resistance by the companies employees and senior management. This made the current CEO Pehr G Gyllenhammar (who was appointed in 1970) resign with immediate effect, marking the end of an era. During the rest of the 1990s the design language of Volvo’s cars changed dramatically, resulting in the S80 of 1998.

In 1999 Volvo decided to sell its car manufacturing business to Ford Motor Company, the reason behind the sale was the high cost of developing new models for a small car manufacturer such as Volvo. The management team believed that the brand could live on as a part of a larger conglomerate. This led to the branching of Volvo Cars and Volvo AB,

where Volvo Cars became part of Ford Premium Automotive Group (PAG) together with Aston Martin, Land Rover and Jaguar. Ford's strategy was to position Volvo Cars, close to the premium segment, a market position better suited for a car manufacturer with the production volume of Volvo Cars. During the beginning of the 21st century Ford, along with the other American Car manufacturers, were losing sales. Volvo on the other hand continued doing well, production increased at a steady rate and several popular new models were launched, including the first generation XC90. Volvo reached new record sales 2004 when 456 000 cars were sold. This in contrast to Ford, which was not faring nearly as well. By 2007 Ford had sold both Aston Martin, Jaguar and Land Rover. Rumours were beginning to emerge that Ford was considering selling Volvo as well. In 2010 The Chinese investment company Zhejiang Geely Hoolding Group had bought Volvo Cars for an, under the circumstances moderate price of $1.8 Billion SEK[8].

Under the ownership of Geely Volvo Cars continued the journey upwards through the segments. After the separation from Ford Volvo Cars made the decision to develop its own engines and a completely new product platform. Great resources were invested into the development of the new SPA – The Scalable Product Architecture – platform. The SPA platform was to function as a foundation for the company's mide-size (60-series) and premium-size (90-series) vehicles. The first vehicle to be launched on the new platform was the 2nd generation XC90, launched in 2014 it became an immediate success, winning prestigious prizes around the globe and exceeding expected sales by some distance. Volvo Cars are currently in the process of launching both the S90 and V90, also based on the SPA platform. During the fiscal year of 2015 Volvo Cars broke its previous sales record, selling over half a million cars for the first time ever (503 127 cars were sold in 2015). The same year the company generated a revenue of 164 043 Million SEK, resulting in an EBIT of 6 620 Million SEK and a net income of 4 476 Million SEK. Volvo Cars had 28 119 employees at the end of 2015, an increase of 7% compared to 2014 (when the company had 26 101 employees).[9]

Throughout the decades Volvo Cars have made some key innovations within the automotive industry, most of them within the area of safety.

- **The Three-point Safety Belt (1959)** – Invented by the Volvo engineer Nils Bohlin, no other safety innovation is believed to have saved as many lives. Volvo waved its patent rights so that all vehicle OEMs could use the invention.
- **Rearward-facing child safety seat (1972)** – An innovation which is used world-wide today, greatly decreasing the injury risk for infants and young kids.
- **The Lambda Sond (1976)** – This little device, no bigger than a finger, is an oxygen sensing probe fitted to the engine. It reduces harmful emissions with up to 90% and is today used in every petrol engine car in the world.
- **Side Impact Protection (1990s)** – By reinforcing cross-members and designing a strong structure of energy absorbing material Volvo greatly improved protection from side collisions, the side protection safety was further improved by introducing the first side airbag in 1994 and inflatable curtains in 1998.
- **Roll-over Protection System – ROPS – (2002)** – As the popularity of SUVs in-

creased Volvo Cars realized the need for enhanced roll-over protection. ROPS includes a sophisticated electronic Roll Stability Control and boron steel reinforced structures.

- **Active Safety Innovations (21st century)** – Volvo is a pioneer within active safety, introducing a flurry of innovation within the area. Innovations include Blind Spot Information System (BLIS), Collision prediction and pedestrian detection, both with full auto brake.

2
Theory

The theory chapter contains different bodies of knowledge which are covered by this master thesis project. In the first two sections product development theory is presented, followed by rotations matrices. After this the software used is presented before the chapter is concluded with basic wheel suspension theory.

2.1 Conventional Product Development

Conventional product development is based on a *Design-Build-Test* (D-B-T) philosophy [10]. Due to the D-B-T philosophy conventional product development is carried out in a sequential manner, which leads to a design process with prolonged lead times and elevated product costs. [11]

The prolonged lead times and the elevated product costs are partly due a phenomena referred to as *The Design Paradox* [12]. *The Design Paradox* revolves around design flexibility, product knowledge and time, the paradox can be described in the following way:

In the beginning of a product development project, the decision maker has high flexibility when making decisions, as no or few constraints are set. However in order to make informed decisions, the decision maker needs a certain amount of knowledge about the product. As the project has just recently begun it is quite probable that this knowledge has not yet been acquired. As the project progresses over time more and more knowledge is acquired, enabling more informed decisions to be made. However, previous decisions, which likely were less informed have imposed design constraints on the product, thus reducing the design freedom, Figure 2.1. This can lead to increased costs later on or even the need to re-take old decisions, which adds further to the lead time and development cost.

The D-B-T philosophy relies heavily on building and testing of physical prototypes the development project becomes both time consuming and expensive. A second issue in the conventional product development process is that design and manufacturing tend to be disjointed. It is not uncommon that the product in question is adjusted to fit the available manufacturing processes after the final design has been released. Moreover, design changes at this stage tend to come with high costs and prolonged lead times and are thus undesirable [11], Figure 2.2 illustrates the cost of making changes at different stages of the development process.

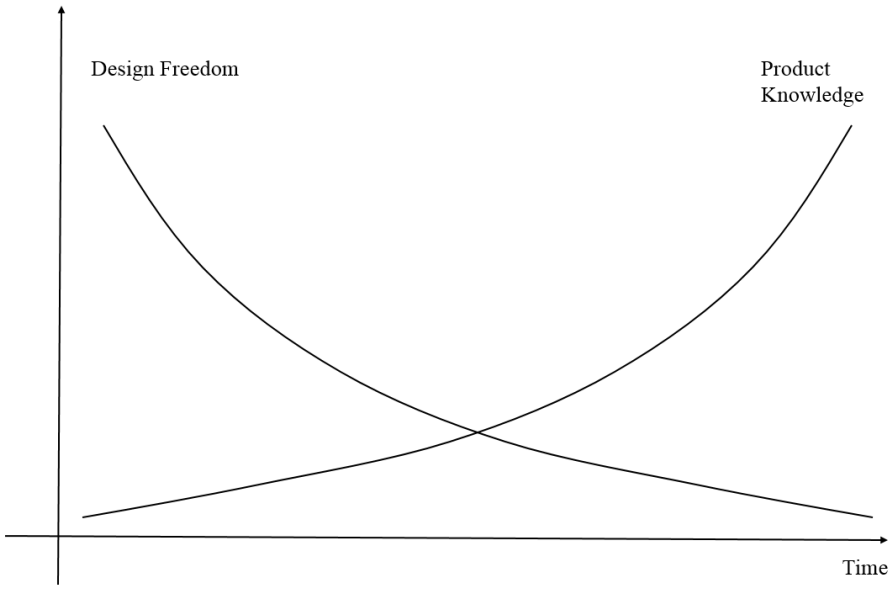


Figure 2.1: Illustrates the *Design Paradox* of product development.

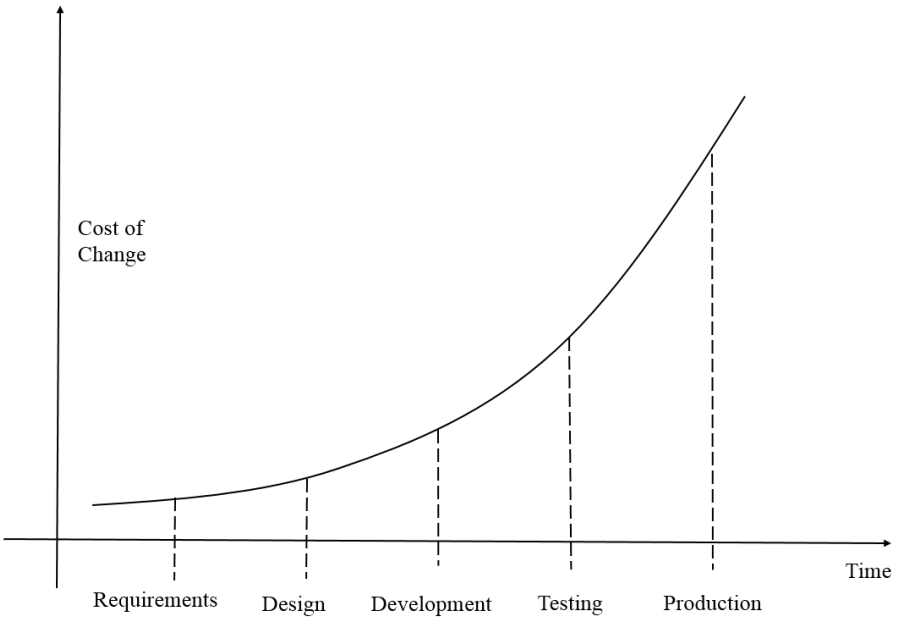


Figure 2.2: Illustrates the cost of implementing changes at different stages in the product development process.

By restructuring the development process companies can save both time and money, Figure 2.3 illustrates just this. The index Cost/ECR (Engineering Change Request) is low during the design stage and peaks during testing. As the cost and complexity of implementing changes in later stages is high the product often becomes delayed. In order to make up for the lost time and deliver according to plan it is not unusual that product quality is compromised. According to Anderson [13] merely 8% is spent during the design stage. However, the decisions made determine 80% of the product lifetime cost. Thus making

informed decisions early on becomes crucial for product lifetime cost, making the "right" decision in the design phase would lead to less ECRs later on, thus reducing product lifetime cost.

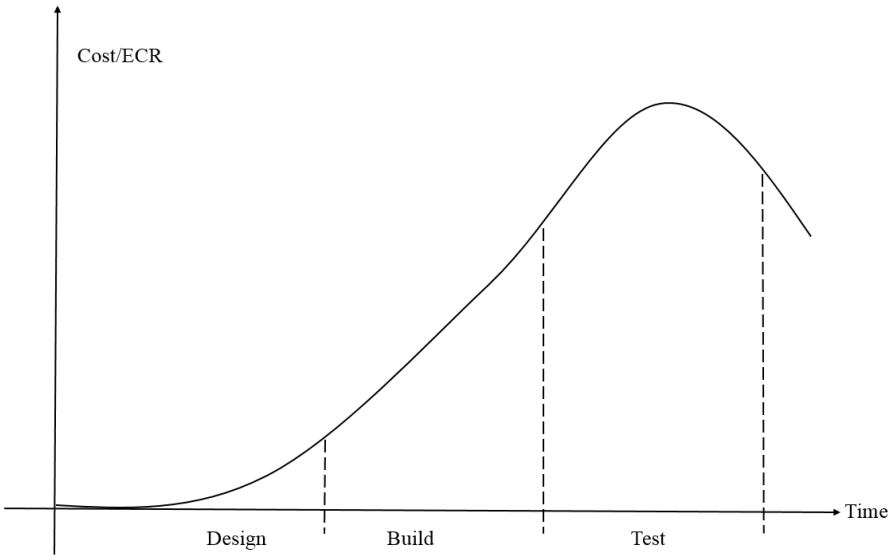


Figure 2.3: Illustrates the Cost/Engineering Change Request in different stages of a conventional design cycle.

2.2 E-design

E-design emphasises the integration of software and IT-related technology in product development. E-design itself is not a process which can replace the conventional product development process, rather it enables the use of concurrent engineering [14] through virtual and physical prototyping with a systematic and quantitative method enabling more informed decision making earlier in the design phase. In addition E-design excels at performance and reliability assessment, as well as improvement of complex, large-scale computation-intensive mechanical systems. [11]

The E-design paradigm enables cross functional teams to simulate and design mechanical products concurrently in the early stages of development. This is done by introducing simulation and design tools earlier in the product development process. By intense use of simulation more product knowledge can be obtained early on, which enables the decision maker to make more informed decisions, thus increasing the likelihood that the "right" decisions are made. This leads to prevention or reduction of the “Design Paradox” phenomena described in Section 2.1, Figure 2.1. Ultimately the implementation of E-design leads to reduced development costs, as simulations can replace physical prototypes, and manufacturing costs, as issues are found and dealt with prior to production. By identifying issues earlier the number of late ECRs can be reduced, making it easier for engineers to deliver according the schedule while not forcing them to compromise on product quality.

The E-design paradigm is heavily reliant on the advancement of computation time. With

faster computation times more hardware tests can be replaced by computerized simulations, thus reducing costs and shortening the lead times even further. This can shift the Cost/ECR distribution, throughout the development process, as illustrated in Figure 2.4. The effect of the *"Design paradox"* can be reduced as more knowledge is obtained in the earlier stages of development, this is illustrated in Figure 2.5.

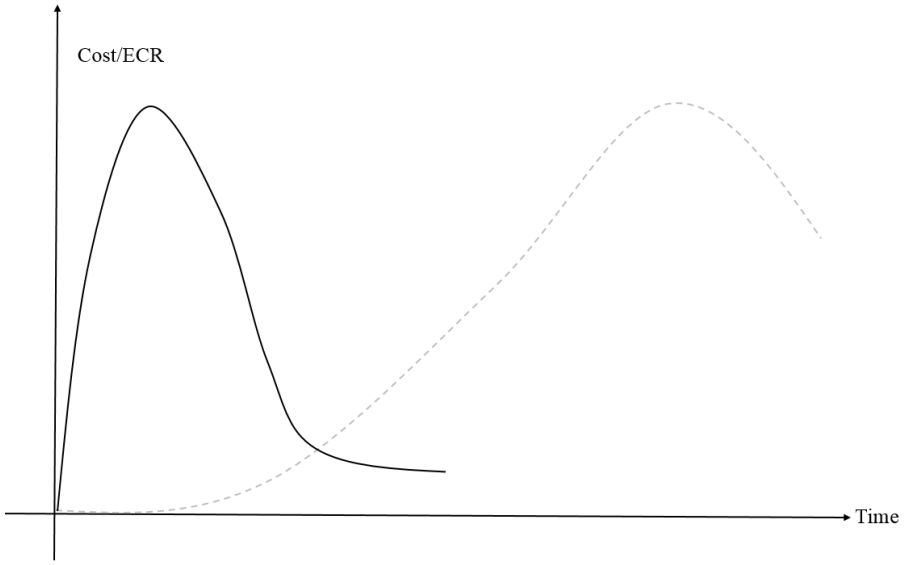


Figure 2.4: Illustrates the shifted Cost/Engineering Change Request in a design cycle when using E-design.

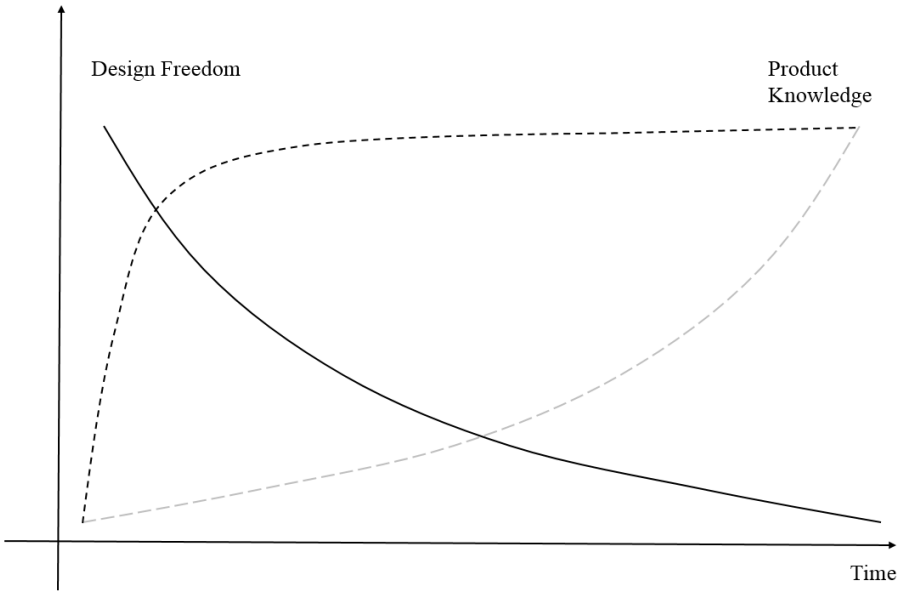


Figure 2.5: Illustrates the shifted design paradox by use of E-design.

2.2.1 The E-design Paradigm

The E-design paradigm is comprised of the ares illustrated within white ellipses in Figure 2.6. The first iteration of a product is often based on the design engineers knowledge and experiences from previous projects. This knowledge and experience is highly desirable to capture and use as support for the current development project, as well as upcoming development projects. The ellipse named *KBE* (*Knowledge Based Engineering*), in Figure 2.6 represents this captured knowledge and can be used by other design engineers when developing new designs, both in CAD and CAE software [15]. Once the design is represented by a CAD-model simulations regarding performance, reliability and manufacturing can be carried out concurrently, throughout the entire development project [11].

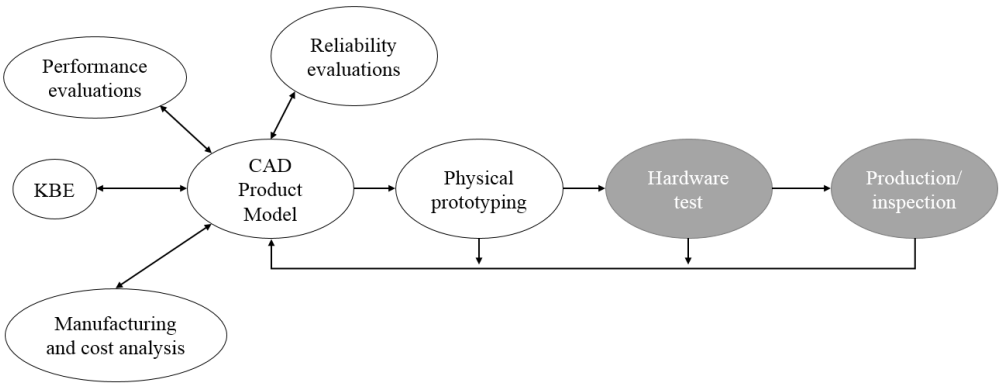


Figure 2.6: The white colored ellipses are the areas which are included in the E-design paradigm

2.2.2 Product Performance Analysis

In order to replace much of the physical testing computerized simulations must be carried out in all areas which are covered by physical testing, this leads to quite a few different types of analysis. Some of which are explained in the list below and fall under the category Performance Analysis in Figure 2.6.

- **Motion Analysis -** Simulations which include kinematics, rigid- and flexible-body dynamics and inverse dynamic analysis. These analysis' can be used to evaluate packaging of a mechanical system, such as the wheel suspension.
- **Structural Analysis -** By using finite elements a mesh which represents the model is created. The number of performance indices which can be evaluated is enormous. Some analysis types are linear static analysis, buckling analysis and fatigue analysis.
- **Fatigue and Fracture Analysis -** Simulations are carried out using finite elements. Fatigue and fractures are a common occurrence in mechanical systems due to cyclic loading.
- **Product Reliability Evaluations -** Evaluation of how often certain events could occur. An event could be how often a certain failure mode will occur.

2.2.3 Challenges

Integrating CAD and CAE software is a complex task which software OEMs have begun working with in recent years. Several companies in various industries have found it more efficient, both time and cost wise, to develop their own interfaces using third party software. The list below summarizes the key challenges which must be overcome:

- **Consistency -** One of the biggest challenges is to ensure consistency, between the CAD model and CAE simulation, throughout the product development cycle. If the consistency is not kept throughout, the simulations will be neither truthful nor useful. [11]
- **Configuration -** Another challenge when creating the model is to ensure that the correct configuration is set. In the case of a kinematic analysis, this would mean ensuring that the joint types, force connections and reference system used match a corresponding physical test. [11]
- **Parametrization -** In order to enable efficient exploration of several design alternatives a parametrized CAD model is required. The CAD model is parametrized by defining a set of dimensions which control the entire model and by establishing relations between these dimensions. [11]
- **Stand-alone Software -** Concurrent engineering requires simultaneous use of CAD and CAE, for design and analysis. Most commercial CAD and CAE software lack the ability to collaborate as they are stand-alone software. One reason for this is the depth required within each field. There is no universal solution which excels in all fields. There are even specializations within both CAD and CAE software [16]. One example of this is ADAMS/Car, described in Section 2.5, which is a CAE software developed solely for automotive analysis.
- **Data Handling -** As most commercial CAD and CAE software are stand-alone data is built up and handled differently in each software. Thus information exchange between different software is a rather complex task, often requiring in-house developed interfaces. [16]
- **Computation Cost -** Depending on how the CAD and CAE integration will be set up the computing cost can be an issue. For example, setting up an autonomous optimization routine will required more computation time, increasing the computation speed would require more CPUs, which in turn would cost more money.[16]

2.3 Coordinate System & Rotations

There are three types of Coordinate Systems (from her eon referred to as CS) which are used in CAD and CAE software; Cartesian, Cylindrical and Spherical [17]. Both ADAMS/Car and CATIA work with Cartesian coordinates as standard, however ADAMS/Car can be configured to work with any of the three. A point in a Cartesian CS is defined by the perpendicular distance to each axis, Figure 2.7.

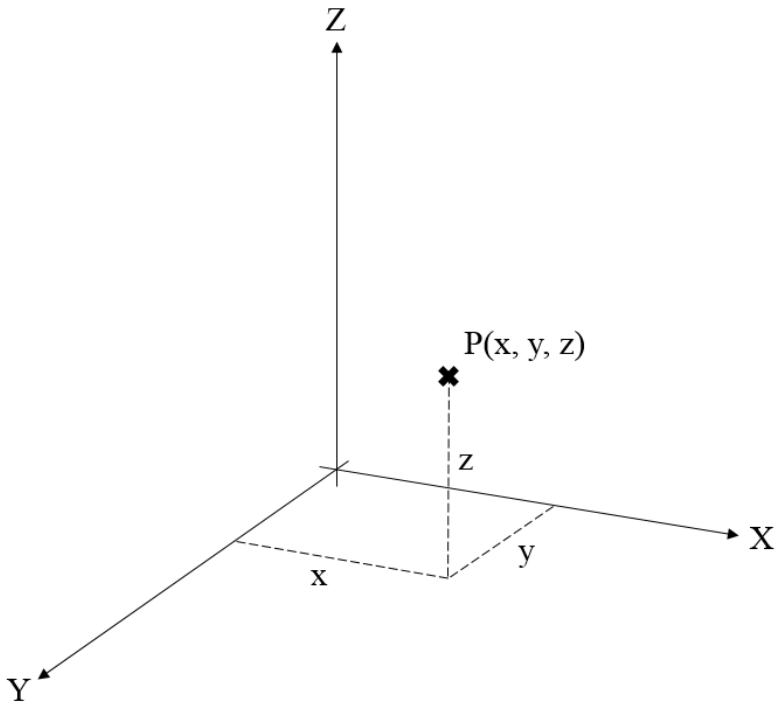


Figure 2.7: Illustrates the position of a point in the Cartesian coordinate system.

In the XY-plane the perpendicular distance to the X-axis is the points position in the Y-direction, while the perpendicular distance to the Y-axis (also in the XY-plane) is the points position in the X-direction. The points distance in Z is defined as the distance from the XY-plane, when working, either in the XZ-plane or YZ-plane. The position of an arbitrary point can be expressed as in Equation 2.1:

$$P = \hat{x} + \hat{y} + \hat{z} \tag{2.1}$$

2.3.1 Rotation Sequences

There are twelve types of approved rotation sequences [18, 19], these twelve can be divided into two groups of six depending on if two or three different axes are used for rotations. Euler angles use only two different axes while Tait-Bryan angles use three different axes. Academics argue about this categorization, some refer to all twelve sequences as Euler angles while others want to classify them as illustrated in Table 2.1.

Table 2.1: Illustrates the twelve approved rotation sequences and how they are categorized into true Euler angles and Tait-Bryan angles. The rotation sequences are classified by the number of axes used during the rotation sequence.

Euler Angles	**Tait-Bryan Angles**
1-2-1	1-2-3
1-3-1	1-3-2
2-1-2	2-3-1
2-3-2	2-1-3
3-1-3	3-2-1
3-2-3	3-1-2

In order to understand the rotation sequences a global CS and a local CS (also referred to as body CS) must be defined. The axes of the global CS are denoted; x, y, z, while the axes of the body CS are denoted; X, Y, Z. Rotations may take place either around XYZ, xyz or both (in the scenario that XYZ and xyz coincide). A rotation sequence which takes place around xyz (the global CS) is called "an extrinsic rotation sequence", while a rotation sequence around XYZ (the local, or body, CS) is referred to as "an intrinsic rotation sequence" [18, 19], the later being the more common according to Boström [18].

Now that xyz and XYZ have been defined the rotation sequences can be explained. Below a 3-1-3 rotation sequence is explained 2.8. This is the sequence which ADAMS/Car uses as standard, the software can however be run with any of the twelve sequences. The Intrinsic 3-1-3 rotation sequence is referred to as an Euler sequence as only the X and Z axes are used for rotation, note however that this includes both x, X, z and Z, i.e. rotations can occur around both the global CS and the body CS.

1. The first rotation (φ) takes place around both Z and z as the two axes coincide, repositioning the x-axis and y-axis.
2. The second rotation (θ) takes place about the new x-axis (x'), repositioning the new y-axis (y') and the z-axis.
3. The third rotation takes place about the new z-axis (z'), repositioning the new x-axis (x') and the new y-axis (y''), these three steps are illustrated in Figure 2.8.

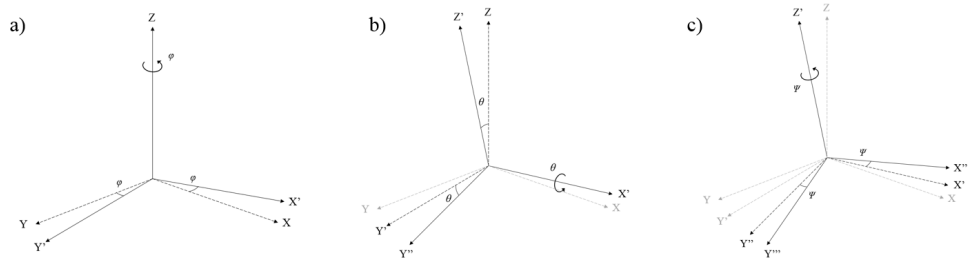


Figure 2.8: Illustrates the order in which rotations take place around each axis in the Intrinsic 3-1-3 rotation sequence.

The Extrinsic 1-2-3 rotation sequence is classified as a Tait-Bryan sequence as all three axes are used during rotation. This is the rotation sequence used in CATIA, Figure 2.9

- The first rotation (φ) takes place around the x-axis, repositioning the y-axis and z-axis.
- The second rotation (θ) takes occurs about the original y-axis (y), repositioning the and the new z-axis a second time (z").
- The third rotation (ψ) takes place about the original z-axis (z), repositioning the new x-axis (x") and the new y-axis (y"), these three steps are illustrated in Figure 2.9.

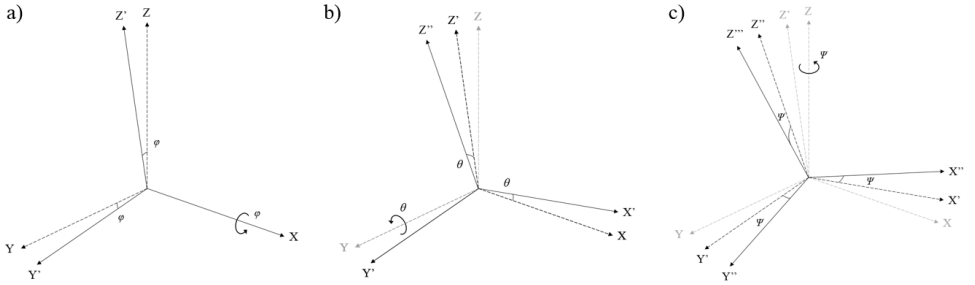


Figure 2.9: Illustrates the order in which rotations take place around each axis in the Extrinsic 1-2-3 rotation sequence.

2.3.2 Rotation Matrices

Rotations are, as mentioned, commonly represented by Euler or Tait-Bryan angles. The mathematical way of expressing the rotation of an object relative to a frame is by setting up a rotation matrix consisting of the three vectors. Thus obtaining a 3-by-3 rotation matrix [19], illustrated in Equation 2.2. Where the vector from the first rotation makes up the first row an so on.

$$RotationMatrix = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \tag{2.2}$$

Every Euler and Tait-Bryan rotation sequence has a unique rotation matrix as the order in which rotations take place matter in 3D space [18]. Two of the most common are the 3-1-3 and the 1-2-3 sequences [18], which are presented below in Equations 2.3 and 2.4. As previously mentioned ADAMS/Car uses the Intrinsic 3-1-3 Euler sequence as default.

$$R_{313}(\phi,\theta,\psi) = \begin{bmatrix} c_\phi c_\psi - s_\phi c_\theta c_\psi & c_\phi s_\psi + s_\phi c_\theta c_\psi & s_\phi s_\theta \\ -s_\phi c_\psi - c_\phi c_\theta s_\psi & -s_\psi s_\phi + c_\phi c_\theta c_\psi & c_\phi s_\theta \\ s_\theta s_\phi & -s_\theta c_\phi & c_\theta \end{bmatrix} \tag{2.3}$$

The 1-2-3 sequence is sometimes referred to as Cardan Angles. This rotation sequence is common in the aerospace industry and computer graphics. The Extrinsic 1-2-3 sequence, Equation 2.4, is the set of Tait-Bryan angles which CATIA is built upon.

$$R_{123}(\phi,\theta,\psi) = \begin{bmatrix} c_\theta c_\psi & c_\theta s_\psi & -s_\theta \\ s_\phi s_\theta c_\psi - c_\phi s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\theta s_\psi \\ c_\phi s_\theta c_\psi + s_\psi s_\psi & c_\phi s_\theta s_\psi + s_\phi s_\psi & c_\theta c_\psi \end{bmatrix} \tag{2.4}$$

2.4 CAD: Catia

CATIA is a CAD & CAE software developed by Dassult Systemes. The software offers an array of workbenches which support engineers in most stages of the product development process. From mechanical design, to structural simulation, to manufacturing.

2.4.1 Wireframe model

The Wheel Suspension unit at Volvo Cars uses a kinematic wireframe model to explore hardpoint-setups in the early stages of development. The wireframe model is built up by hardpoints, which are defined joints positioned relative to the global axis system. Depending on where they are positioned the DOFs (Degrees Of Freedom) vary, some are completely locked while others have six DOFs. The wireframe model is thus a simplified model of the wheel suspension, referred to as an undressed model as no component

geometries are included, Figure 2.10.The model allows design engineers to explore multiple concepts without investing too much time and effort, by allowing for quick adjustment of hardpoints without the need to modify any dressed geometry. For instance hardpoint 3, the joint between the LCA and Subframe, has the initial position (x,y,z), to explore different configurations design engineers can simply move this point to (x_i,y_i,z_i) and preform a new kinematic analysis.

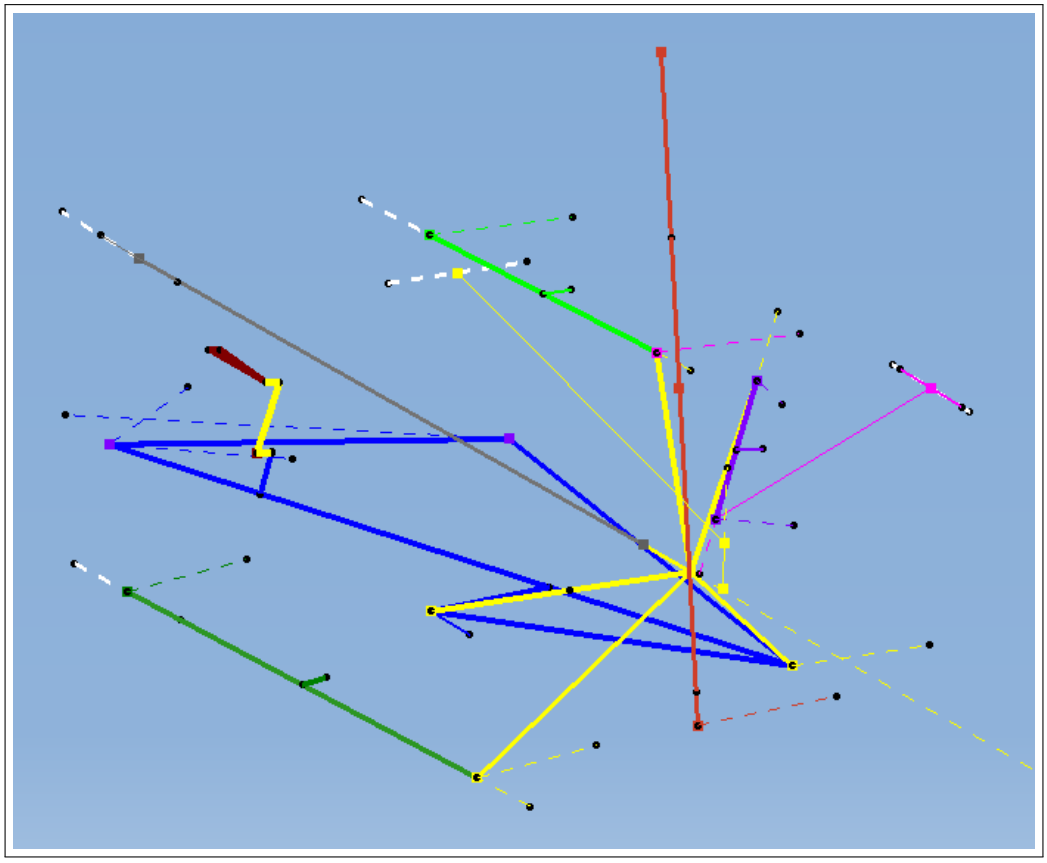


Figure 2.10: Illustrates the wireframe model of the rear wheel suspension of the SPA platform in the S90 setup, courtesy of Volvo Cars.

2.4.2 Kinematic model

A kinematic model, in CATIA, is a model which describes how a system moves and the components of the system relate to each other. In the initial development of a mechanical system it is often unclear how the system will behave and perform, the components of the system may clash and thus interfere with the overall system performance. The kinematic model allows design engineers to study the system and its behaviour in order to predict clashes and and evaluate system behaviour [11, 20].

Figure 2.11 illustrates the working procedure for setting up and running analyzes with a kinematic model in CATIA. In the first two steps, *Product Model* and *Constraints*, the design engineer positions and locks the components in their initial position.

Once the components are connected joints between them need to be defined, the compo-

nents will behave differently when motion is applied depending on which type of joint is used. Laws such as displacement limits, accelerations and other behaviour must also be defined. Joints are set in the third step, *Mechanism* and behavioural properties in the fourth step, *Laws*.

Step five, six and seven, *Simulation*, *Replay* and *Analysis* consist of setting up displacement values for the simulation, running the simulation and creating a replay file. The replay file records the motion of the simulation, which enables it to be saved and replayed for further analysis. [20]

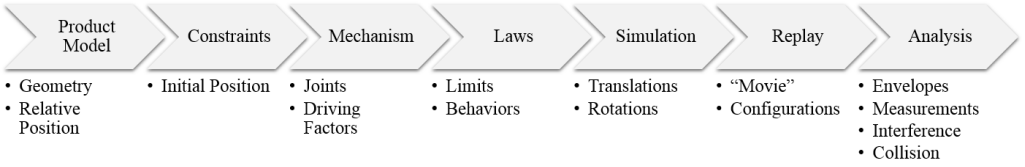


Figure 2.11: Illustrates the work flow for setting up and running a kinematic analysis on an assembly in CATIA V5.

2.4.3 Replay

Replay is an application within CATIA which allows the design engineer to create a "movie" from a specified set of motions. There are two ways to create a Replay in CATIA. The first, and most common way, is through a setting up a mechanism and simulation [21], this process is described in Section 2.4.2. The second way is through CATIA's APIs, which can be accessed by running a script.

The "Create Replay" script consists of two main parts. The first part of the script is handles declaration of which parts and products to include in the replay file. The second part contains all the specified movements, specified for each timestep. Movements are declared trough a combined rotation and translation matrix. CATIA operates using a 1-2-3 rotation matrix, which is described in Section 2.3.The structure of the "Create Replay" script is visualized in appendix B.

Important to note is that, when the movement is executed, its not the virtual part object that is moved and rotated. Instead it is the part objects CS which is moved, this in turn moves the part object. So when a part moves 100mm from its origin, it is the axis system which moves 100 mm from its origin. In order to move the part an additional 100 mm the CS must be moved 200 mm from its origin, i.e the distance a part object is to be moved does not operate with Δ-values but with absolute values. See Figure 2.12 which illustrates how translations are handled.

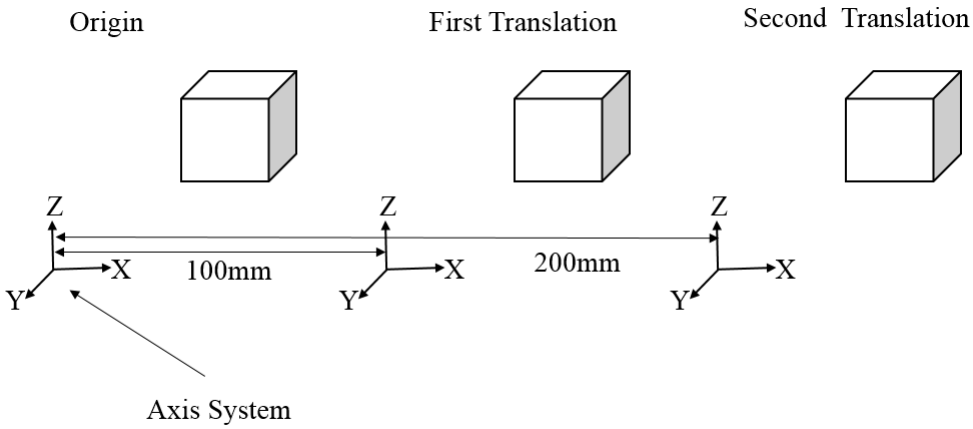


Figure 2.12: Illustrates how translations are handled in CATIA V5.

Rotations are handled in the same way. That is, the object CS is rotated relative to its original position. Figure 2.12 illustrates how an object's axis system is rotated around its Y-axis, relative to its origin, with the angle θ.

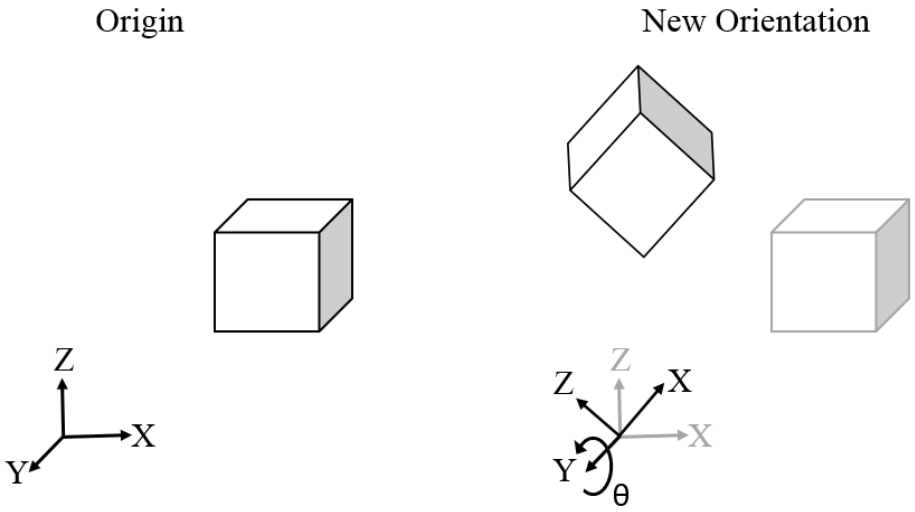


Figure 2.13: Illustrates how rotations are handled in CATIA V5.

2.4.4 Envelopes

An envelope is a representation of all positions which a component takes during a certain set of motions, i.e. the design space of that component. An envelope is usually visualized by a surface model in CAD software but may also be visualized as a diagram or a set of matrices. Envelopes are most commonly used for interference analysis and clash analysis [22]. In the context of the wheel suspension the most common envelope is that of the wheel. The wheel envelope is used to visualize the wheels movement during jounce, rebound and steering motions in combination with drive events such as break-in-pot-hole and drive-over-curb. A surface geometry of the wheel envelope is created, Figure 2.14,

which allows the design engineer to check for interference and collisions within the wheel housing [23].

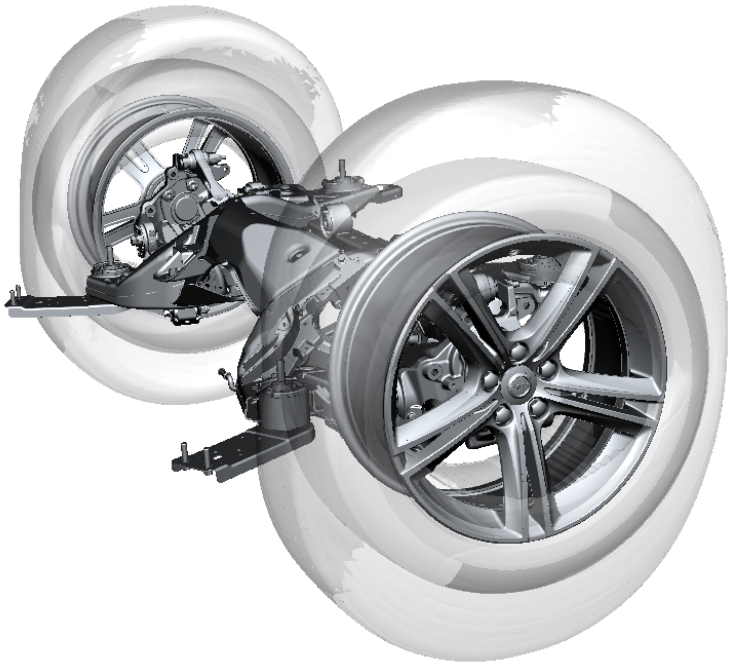


Figure 2.14: Illustrates a wheel envelope from a parallel wheel travel jounce and rebound simulation.

2.5 CAE: ADAMS/Car

ADAMS (**A**utomated **D**ynamic **A**nalysis of **M**echanical **S**ystems) is a multi-body dynamics simulator developed by MSC Software®. ADAMS/Car is a package within ADAMS which enables virtual prototyping of cars. The package includes a number of modules which enable testing of the full virtual vehicle, both at system level and sub-system levels [24].

2.5.1 Body Types

ADAMS/Car offers four different kinds of parts when building a template.

- **Rigid Bodies -** Cannot deform, have mass and inertia properties, are movable.
- **Flexible Bodies -** Can deform when submitted to loads, have mass and inertia properties, are movable.
- **Ground Part -** This part is static, it defines the CS and global origin, it is also used as a reference when calculating accelerations and velocities. The ground part must therefore exist in all models.
- **Mount and Switch Parts**
  - Mount Parts are mass-less "place holders", which are replaced by other parts in an assembly.

  - Switch Parts are also mass-less, they act as switches for connections. By changing the switch part one part will connect to another.

Flexible bodies are preferable, over rigid bodies, whenever component flexibility is believed to influence the system dynamics. Some of the implications for this thesis stems from the way the two component types are constructed and how component movement is logged. The movement of a rigid body is logged from the COG of the specific part, while the movement of a flexible body is measured by logging the movements of each node which the flexible body consists of.

2.5.2 Logging Data

When running a simulation in ADAMS/Car it is often of interest to log the translations, rotations, velocities, forces and frequencies which certain components are subjected to. This is done by using *Markers* and *Requests*. A marker works like a sensor which logs any or all of the above mentioned parameters for a specified component, node within a component or set of nodes within a component. Logging data of individual nodes can be of interest when dealing with flexible bodies, the same goes for logging data from a set of nodes, thus obtaining a mean value.

A *request* is used to extract the logged data from the marker, making it available for post-processing operations, either within ADAMS/Car post processor or for use in other software through the use of .txt-files, .xml-files or the .res-file, which is introduced in the next section. A request can be defined in three different ways:

- **By using type and markers -** The user specifies if the request is to contain displacements, velocities, accelerations or forces. Further the user specifies the I marker (which marker tog log data from) and the J marker (which marker to log data relative to).
- **By defining a subroutine -** The user may define a subroutine outside of ADAMS/-Car, e.g. in C, C++, Python or other programming language. The request calls the subroutine which may transform the logged data to a certain format or export it in a certain way.
- **Using function expressions-** The user may specify an arbitrary function which transforms the logged data according to the users specifications.

The standard output of a request is made up of eight slots of which four are reserved for translational parameters and four for rotational. Two of the slots are reserved for the translational and rotational magnitudes respectively while the remaining six can be defined as the user pleases. The user also has the possibility to modify all eight output slots to contain other data, such as accelerations, angular velocity or inertia to name a few.

The process of placing a marker and defining a request is quite straight forward, Figure 2.15. All the work is done in the template builder mode, the user opens the templates which make up the assembly of interest and adds markers as explained above, thereafter the

user defines requests which call the markers, also explained above. Finally it is important to save the changes in the template and close and re-open any active assemblies for the changes to take place.

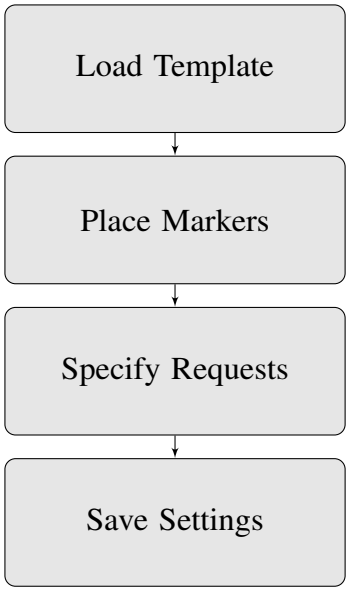


Figure 2.15: Illustrates the workflow of modifying the ADAMS/Car templates such that motion extraction is enabled.

2.5.3 ADAMS/Car .res-file

ADAMS/Car produces, among other files, an output file of .res format. This file contains information about all components in the model and their behaviour during the simulation run. The .res-file begins by stating general information such as simulation date, solver, user, number of timesteps and units used. Thereafter the entities (which make up the model) are defined. Entities contain different information depending on their type, however every entity contains the following information classes:

- **Name -** Entity name can be assigned by the user in the template builder.
- **Entity -** States the original name, based on the structure in which the entity lies, if **Name** is not changed **Entity** will be the entity name as well.
- **Entity Type -** States the type of entity, e.g flexible body, part or request, to name a few.
- **Object ID -** Numbers the entities, numbering is performed for each entity type.

Entities which contain motions also have an information class named **Component**. This class contains a certain motion, be it a translation, rotation, acceleration, etc.... Each component consists of three sub-classes:

- **Component Name -** Self Explanatory, the name may be changed by the engineer who builds the template.
- **Units Value -** Contains unit of the logged information.
- **ID -** The ID states which position a certain components motion has for each timestep. To clarify, if a component, stating the translation around the x-axis for a certain entity, has the ID = 5629 then the motion will be the 5629^{th} value stated for each timestep.

The last, and largest part, of the .res-file contains the timesteps for the simulation, stated in order from 1 - *n*. Each timestep contains only the numerical value of each component for the given timestep, thus identifying a specific value can be tricky as there may be tens of thousands of component values. In order to put the size of a .res-file into context a simplified simulation run of one drive event with 10 timesteps takes up approximately 1100 A4 pages, containing approximately 12000 components. When "real" simulation runs are done the number of timesteps are increased tenfold, or more.

2.6 Wheel Suspension Theory

In this section some key parameters for wheel suspension design are presented. For a description of different wheel suspension types see Appendix A.

2.6.1 Basic Concepts

The Wheel Suspension is a vital system when setting the characteristics of a car, as it acts as the interface between the road and car body it influences both ride comfort, handling and safety. In order to understand how a wheel suspension system is designed there are some basic concepts which must be presented.

When defining a system such as the wheel suspension a coordinate system is required. Most commonly used is the ISO 8855 coordinate system, Figure 2.16. Oriented from the driver position the x-direction is facing forward (the longitudinal direction of the vehicle), the y-direction to the drivers left side (lateral direction) and the z-direction heading upwards (vertical direction) [25]. The origin of the coordinate system has no fixed position and may be placed anywhere along the xz-plane.

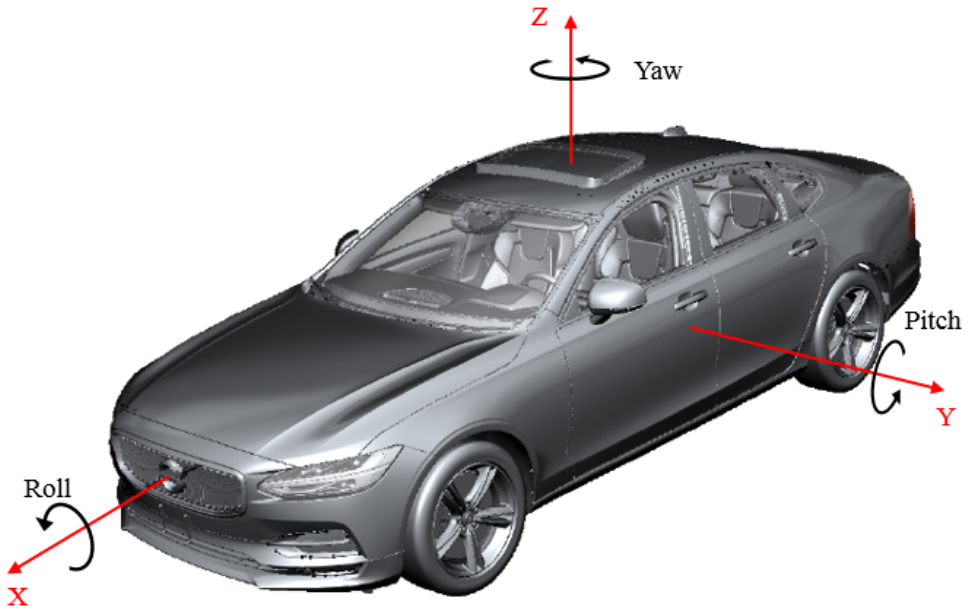


Figure 2.16: Illustrates the vehicle coordinate system as defined in ISO 8855 and DIN 70000 [25]

Rotational motions around these three axes are referred to using separate names. Rotations around the x-axis are referred to as roll, rotations around the y-axis as pitch and rotations around z as yaw.

Pitching motions (rotations around the y-axis) can be divided into dive, squat and lift. They take place during breaking an acceleration. Dive occurs when the nose of the car dips during breaking, simultaneously lift occurs at the back of the car. Squat occurs during acceleration, when the back of the car "sits down", simultaneously lift occurs to the front of the car.

Rolling motions take place mainly during cornering but are present in all situations where a lateral load is applied. Rolling motions occur around the vehicles roll-axis, which is obtained by drawing an imaginary line between the roll-centers of the two wheel pairs. The respective roll-centers are defined by two intersecting lines, one from the moment center of the wheel (seen from the rear) to the point of contact with the ground and the other through the center of the vehicle (draw vertically), thus the roll-center is positioned at the intersection between these two lines, Figure 2.17. However, note that as the center of the wheels moment is used to obtain the roll-center it will vary depending on the position of the wheels, as the center of moment shifts during e.g. jounce & rebound movements.

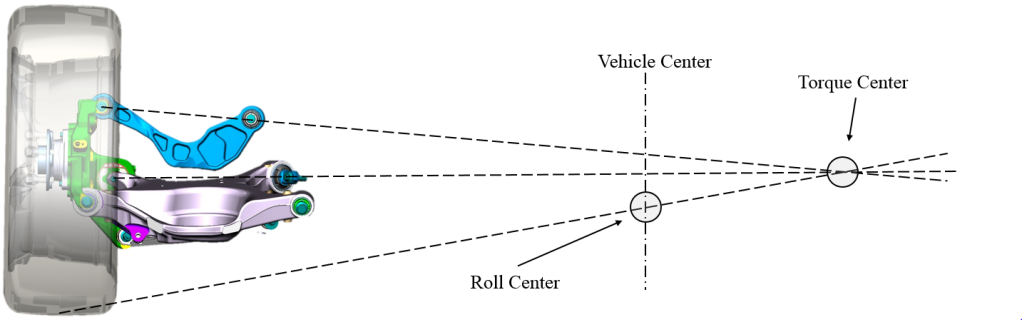


Figure 2.17: Illustrates how the roll center of the vehicle is located.

Yawing motions occur when the car rotates around its vertical axis, a scenario where this happens is be when the driver loses control and the vehicle starts spinning.

All of these motions are desirable to control, both from a handing, ride and safety perspective. The degree of motion which is desired varies depending on the targeted market segment, customer base and vehicle type. One factor which affects both pitching and rolling motions is the vehicle's center of gravity (COG), by lowering the COG both dive, squat and roll can be reduced, which leads to better handling and thus also increases passenger safety [26]. Other factors which affect rolling and pitching motions are wheelbase and track width. The wheel base is defined as the longitudinal distance between the midpoints of the two wheel pairs, while the track width is the lateral distance between the midpoints of two wheels. Apart from counteracting pitching motions the wheel base also affects passenger space and handling. A shorter wheel base results in less room for passengers, but also increases maneuverability, e.g. when cornering or parking, and generally reduces weight and cost. By increasing the length of the wheel base there is more room for passengers and the ride is improved. The track width affects the handling, passenger space and aesthetics of the vehicle. By increasing the track width roll is reduced, thus increasing handling. Increasing track width also gives the car a more robust and luxurious appearance. Narrowing the track width leads to less stability, less packaging space and less room for passengers, therefore it is, in most cases, desirable to keep the wheel base as wide as possible.[26]

2.6.2 Wheel Suspension Parameters

Even though steering is an important part of the wheel suspension it will not be treated in this report. The reason for this is the limitation to focus on the rear wheel suspension, where no steering is present. In order to meet the demands of today's customers the wheel suspension must allow the wheel to move along a complex 3D trajectory during jounce & rebound. Vertical movements alone would not provide the necessary level of ride quality and handling. According to [26] the following five parameters are required to describe the wheel trajectory:

Camber angle γ : The camber angle is the angle between the wheel center plane and the plane perpendicular to the ground plane, Figure 2.18. The camber angle affects lateral dynamics and by setting a slightly positive camber angle wheel wobble can be suppressed,

however this is done at the expense of resistance to lateral forces. Race cars usually have a larger negative camber angle than ordinary cars in order to cope with cornering, as great lateral forces are applied when cornering at high speeds. The drawback with a high camber angle is asymmetric tire wear, due to the angle of the tire.

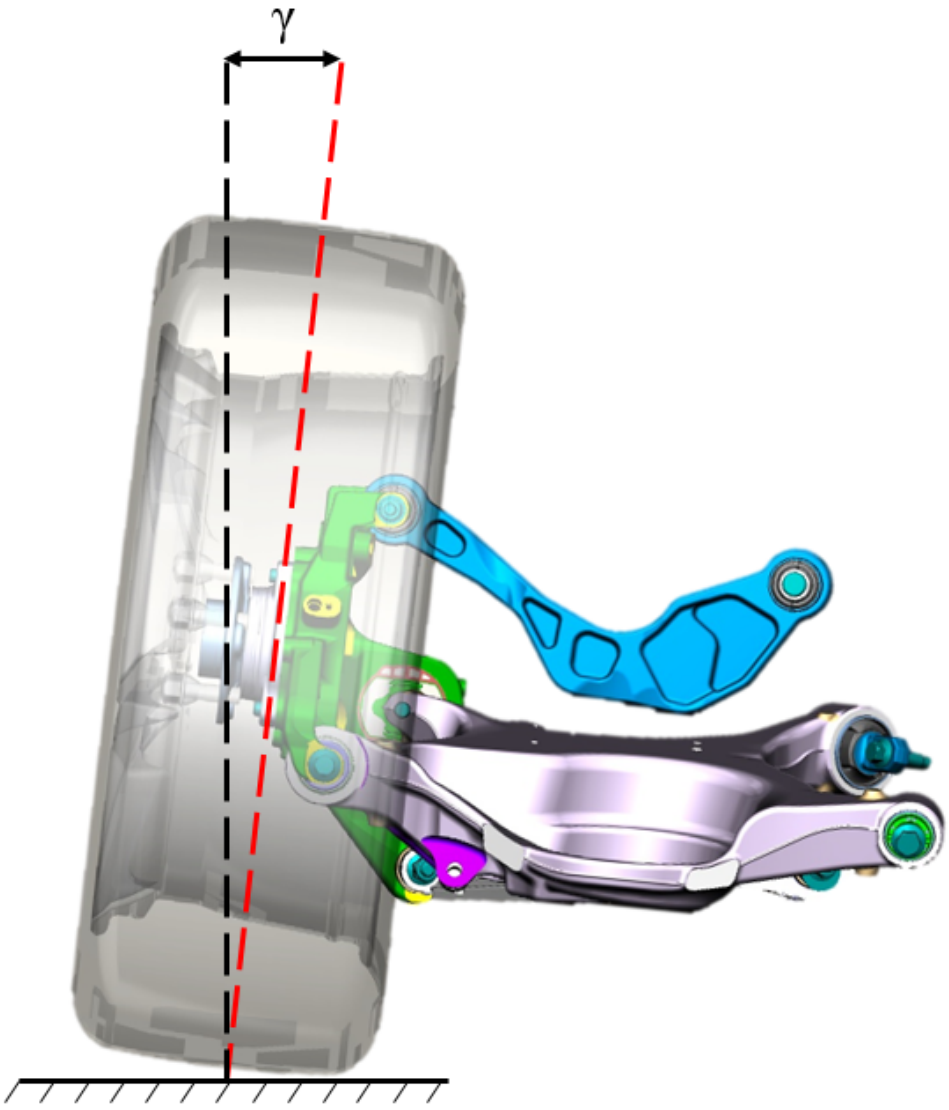


Figure 2.18: Illustrates the camber angle.

Roll Center: The point in the yz-plane around which the car rolls. If the roll centers of the two wheel pairs differ the roll center will be a mean between the two. Higher roll center reduces body roll as the distance between the roll center and COG is reduced, while a lower roll center reduces camber and track width changes during jounce and rebound.

Toe angle δ & toe angle change: The toe angle is defined as the angle between the wheel center plane and a plane parallel to the xz-plane which runs through the wheel center, seen from above, see Figure 2.19 [25]. A slight toe angle is used to control handling, as the toe angle influences straight line driving stability, cornering behaviour and suspension tuning.

Manufacturers also use the toe angle to compensate for elastic deformation while driving.

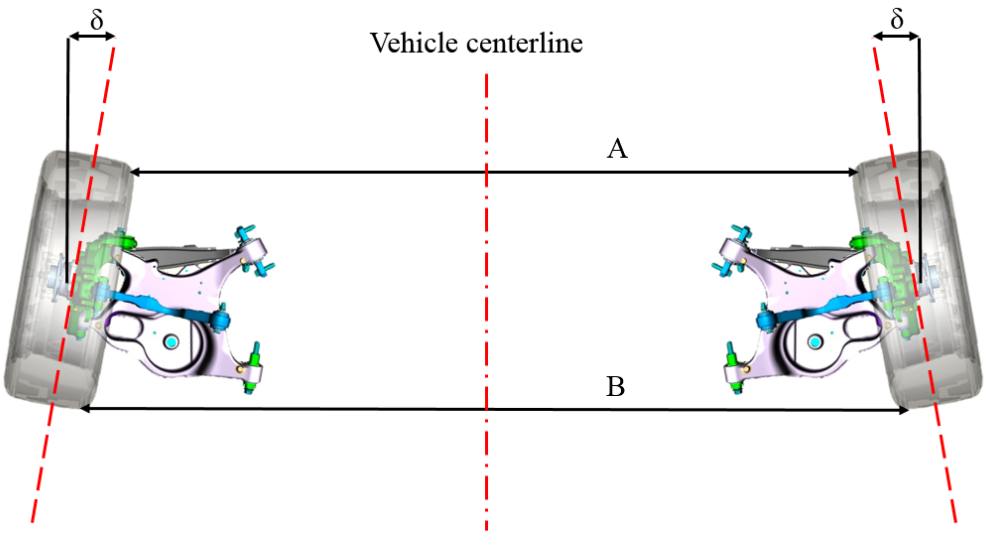


Figure 2.19: Illustrates the toe angle, toe in when A < B, toe out when A > B. Note that the toe angle in this picture is greatly exaggerated for educational reasons.

Camber angle change: During jounce and rebound the camber angle changes, thus influencing all the previously mentioned areas which the camber angle affects, it is desirable to control this change to the greatest extent possible. A theoretical example is straight line driving vs. cornering, during straight line driving a slightly positive camber angle suppresses wheel wobble, however a slightly negative camber angle improves the cars ability to cope with lateral forces which are present during cornering.

3

Methodology - Interface Design Between CAD & CAE

This chapter is an integrated methodology and "execution of work" chapter where each challenge faced is treated separately. The chapter begins by presenting the current working process for packaging analysis of critical components in the wheel suspension. Thereafter the Pre-study is presented, followed by the Requirement study. The section on the Concept Development phase is initiated with a section describing the overall working methodology before treating the challenges faced, finally the chapter is concluded with a section about the Detail Design phase.

3.1 Current Working Process

In order to understand where improvements could be made, the current working process for creating a packaging volume of critical components was mapped. Meetings were held with involved design engineers where the current process was presented and discussed, from these discussions the working process, visualized in Figure 3.1, was extracted.

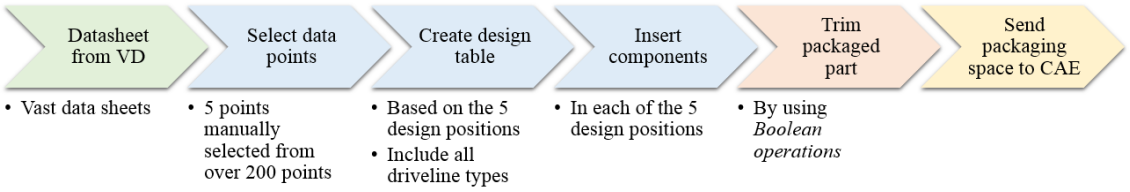


Figure 3.1: Illustrates the steps of the current working process for packaging of critical components.

The process was carried out manually using generic rules, or tolerances, based on measurements and analyses, meaning that there were opportunities for improvements to be made both with regard to accuracy but also with regard to lead times. The data sheets received from the Vehicle Dynamics department had to be produced by customized runs where the model had to be modified in order to only include rigid bodies. Further the data from the analysis had to be summarized and visualized in an excel spreadsheet, which meant post-processing work for Vehicle Dynamics. A clear advantage would be to be able to use a compliant model (include flexible bodies and bushing stiffness) and read data from

a standardized file format such as the .res-file which is automatically produced for every analysis, this way no post-processing work would be required from Vehicle Dynamics.

The next three steps; *Select Data Points, Create Design Table* and *Insert Components* require the largest portion of time. As the data sheet from Vehicle Dynamics contains all specified drive cases for creating a packaging volume, for every drive line configuration, the content is quite vast. Manually selecting a number of data points from the data set is a time consuming task, adding to the amount of work is the fact that the task must be repeated for each drive line configuration. The selected data points are summarized in a design table, which is also set up manually. The design table is then used to acquire the location of each component at each measure point. This (inserting components) is the most time consuming step, a solid geometry representing each neighbouring component at each measure point must be created and inserted into the model. Once all components have been inserted for all five data points the component which is to be packaged can be treated. Those components which intersect with the component to be packaged are simply subtracted from it, producing the available packaging volume as a result. By automating this process the steps could be integrated in such a way that they require less time and effort, while producing a more accurate result.

By supplying a tool, either within ADAMS/Car, CATIA or a third party software, which can read the result from ADAMS/Car, read the wheel suspension model from CATIA and allow motions to be mapped to components in the model improvements could be made within all three areas mentioned above. The time and effort required is reduced as the amount of manual work required is reduced. The accuracy is also improved as the generic rules can be replaced by actual motions transferred from ADAMS/Car.

3.2 Pre-study

The pre-study was initiated by a market review where the purpose was to gain an understanding of issues faced within the area of CAD & CAE integration as well as the state-of-the-art within the area. The second part of the pre-study involved learning the necessary software, this included specific CATIA modules and navigating in ADAMS/Car.

3.2.1 Market Review

Both scholar articles, documents from Software OEMs and internal documents within the area of CAE & CAE integration were studied. This is a field which is currently experiencing rapid development. Two key drivers behind this is the need to reduce both development lead times and development costs [11], as explained in Section 2.2. To put the rapid development into context Waterman [27] describes how the ratio of engineers doing design or analysis has gone from six design engineers for every engineer doing analysis to 1.2 engineers doing analysis for every engineer strictly doing design. This means that more design engineers are performing analysis, although at a fairly simple level. The more complex analysis problems are left for the analysis engineers.

So far much effort has been put on the transfer of CAD geometry to CAE software and

how the data is handled to minimize information loss [11][28]. This process is handled by a so called *"translator"* which could be a stand-alone software or a module within a CAD software. The translator converts the CAD geometry to a neutral file format such as STEP or JT. Issues faced in this process range from more general cross-software communication issues such as information losses and data structure compatibility, to more specific issues such as the simplification of complex geometry to enable meshing and analysis [28].

The process of data transfer from CAE to CAD is a less explored area where Software OEMs just recently have begun developing solutions. There are however OEMs, both within the automotive industry and other manufacturing industries who have developed in-house solutions for this type of data transfer. The solutions are either integrated in the CAD & CAE software or developed using a third party software. The type of OEMs doing this are considerably larger than Volvo Cars and thus have the resources to develop their own software, tailored to their specific needs.

3.2.2 Software Study

Training course material from Volvo Cars internal CATIA education was used to get acquainted with the kinematic workbench. The training included setting up and running a kinematic analysis, which included modelling of components, setting constraints, joints and laws. Thereafter specifying the desired motions and creating a replay-file for further analysis. Internal course material for ADAMS/Car was also used, in combination with expert consultation, to gain an overview of the necessary functions in ADAMS/Car. This involved setting up markers, requests, running analyses and handling the results. While executing the training exercises some features where identified which would need to be considered when designing the application.

First of all the body CS', in CATIA, are placed in the global origin by default but can be repositioned. However, a replay file is designed in such a way that all part CS' are placed in the global origin. Consequently rotations will be amplified to an extent, depending on how far away from its CS the body is placed. The placement of the body CS' must be considered when defining what data to extract from ADAMS/Car and how to extract it. I.e. the extracted data must match the movement of the body CS for a certain motion of the corresponding part. The reason for this is that a body is moved not by moving the body itself but by moving its CS.

The fact that kinematics are involved means that great care must be taken in how motions are transferred between software. Rotations will need to be carried out in a certain sequence which matches the rotation sequence of the software, the theory behind this issue is described in Section 2.3.

It will also be of importance that a single component can be assigned only one motion, otherwise CATIA will not be able to create a replay-file. It should however be possible to assign one motion to several components, if desired.

3.3 Requirement Study

By the time that the requirement study was initiated it had become clear that the the most efficient way of developing a working methodology would be to develop an application which would automate cumbersome parts of the current working process. As the process of transferring data from ADAMS/Car to CATIA is both time consuming and has a low degree of automation this part of the process was chosen. There are various commercial translator software available which can connect CAE and CAD software in some way. Either through transferring data or handling the analysis in its own environment. Purchasing any of these software would involve corporate politics at the highest level and thus have large implications for Volvo Cars. Moreover purchasing a third-party software was not an alternative for this thesis as one limitation was to use currently available software licences. Instead three different environments where the application could be developed were identified, as seen in the list below:

1. **ADAMS/Car**.
2. **Third-party software.**
3. **CATIA V5.**

.
The environment selection was based on a set of initial requirements established in collaboration with the project supervisor at Volvo Cars, Daniel Molin [23]. The initial requirements stated that the application must:

1. Allow for quick implementation.
2. Utilize the competence of design engineers within the department.
3. Not differ radically from the current working procedure, in terms of roles and tasks.
4. Add minimal work for the Vehicle Dynamics Department.
5. Preferably avoid increased costs due to new software licenses.

Developing the application within ADAMS/Car would mean that information handling would be concentrated to one software, as wheel suspension movement during drive-events is simulated in ADAMS/Car. This would also mean that the precision would be high as no data conversion is required. However, it would also add to the workload of Vehicle Dynamics, alternatively require design engineers who are used to working in CATIA to learn a fundamentally different software. Moreover this would require additional ADAMS/Car licenses which come at a high cost. Development in an ADAMS/Car environment would thus not fulfill requirements 2, 3, 4 and 5.

Using a third-party software for data transfer, e.g. Matlab or Python, would enable utilization of specific features, e.g. Matlab's excellence in solving mathematical equations. It would also allow for wide scale implementation across the company as these software can be integrated with most others. However, design engineers would need to learn new software and it is quite likely that more licences will need to be purchased, thus violating requirements 2 and 5.

The third alternative is to develop the application within CATIAs VBScript environment. This would mean that the data transfer is handled by the design engineers in the Wheel Suspension department who will be working in an environment which they are familiar with. thus fulfilling requirements 2, 3 and 4. Working in CATIA also means that the required licences are already in place and that quick implementation is possible, thus fulfilling the remaining two requirements (1 and 5).

As using a VBScript environment in CATIA fulfills all the requirements it was established that the application was to be developed in this environment. The next step was to set up a requirement specification and formulate the requirements. The layout of the requirement specification was influenced by Ulrich & Eppinger [29] and is based on seven columns with information:

- **Requirement Area -** Lists the area which the requirement is derived from.
- **Requirement -** States the title of the requirement.
- **Importance -** Rates the importance of the requirement on a scale from 1 to 5.
- **Demand/Wish -** States if the requirement is a demand (D) or a wish (W). Demands must be fulfilled while wishes add extra functionality and increase customer satisfaction.
- **Description -** Contains a short description of the requirement, where necessary.
- **Target Value -** States just this, where applicable.
- **Verification Method -** States how the requirement will be evaluated.

The requirement areas were decided through dialogues with involved parties including design engineers, analysis engineers and representatives from the IT department. Based on the output from these dialogues the application was broken down into functional areas. The functional classification was selected as most of these functional areas can be developed separately, with the exception of maintenance requirements which are implemented throughout. An alternative could have been to classify requirements by user group, however this approach would not fit the working approach with separate sub-system development as well. The functional areas from the breakdown are shown in in Appendix C and the list below:

- Data Extraction from ADAMS/Car
- Data Transfer
- Data Insertion into CATIA V5
- User Interface
- Post-processing
- Maintenance

Requirements were also gathered through close interaction with future users, both in

the form of informal "hallway" meetings, formal meetings and a focus group meeting. The focus group meeting was structured as such that it begun with a short presentation of the planned application and its functional areas. Thereafter each area was discussed separately through an open dialogue, the authors functioned as moderators while also taking notes. In order to ensure that requirements from all user groups were gathered there were representatives from Vehicle Dynamics, The Wheel Suspension Group and IT-support present. The input from all of the meetings was analyzed and compiled into the requirement list seen in Appendix C.

3.4 Concept Development

As this is a software oriented development project there are certain aspects which differ from a more conventional mechanical design based development project. One of the opportunities when developing software is the possibility of increasing the speed of design loops. This is possible due to two aspects. First off, writing and compiling code takes considerably less time than designing and simulating a set of load cases for an arbitrary component or system. Second, as no physical prototypes need to be built for testing lead times are reduced while no capital resources need to be tied up in producing and building of prototypes. This means that the number of design loops, including testing with quick n' dirty prototyping, can be increased greatly.

As design loops become shorter and integrate testing to a greater extent a structured work methodology which allows for this is required. The conventional product development process as described by Ulrich & Eppinger [29] has to some extent, influenced the overall structure of this master thesis, Figure 3.2. This is however a linear process, not very well suited for structuring the fast design loop iterations.

Figure 3.2: Illustrates the working procedure which has been used during this master thesis.

A method more suited for this task is the V-model [30], this method is based on a product breakdown using a top-down approach, sub-systems are then designed separately in integrated using a bottom-up approach. Applying the V-model to the concept development phase of this master thesis therefore meant that the application could be broken down at multiple levels. Sub-systems could then have separate design loops and be integrated when complete, Figure 3.3 illustrates the adaption of the V-model to this project. Initially Matlab was used for proof of concept during sub-system development, thereafter the sub-systems were developed in CATIA V5's VBScript environment.

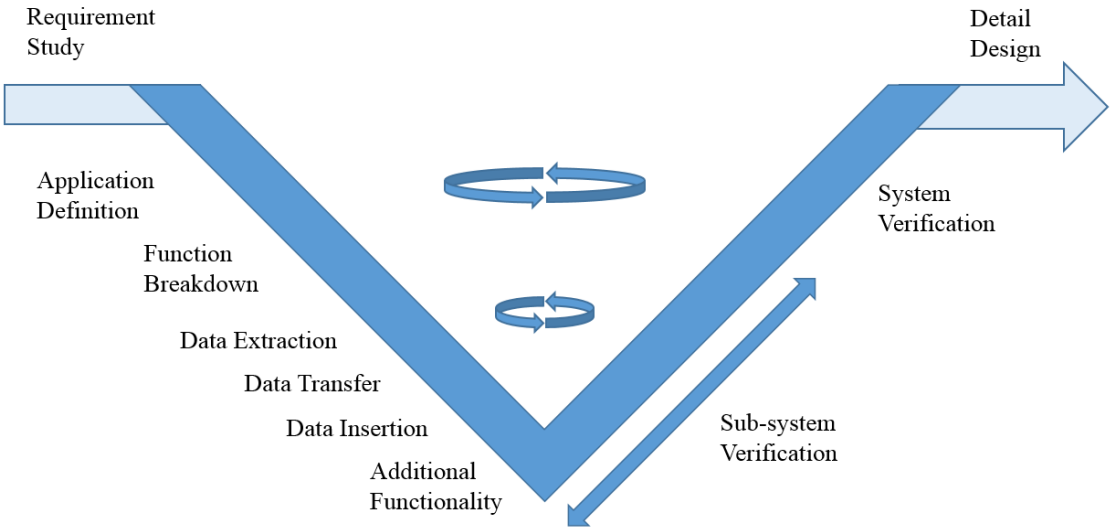


Figure 3.3: Illustrates the breakdown of the application according to the V-Model [30]. This approach was used during the concept development phase of the project.

The system, or application, breakdown was based on the functional areas established during the requirement study as these areas were easy to assign requirements to. Naturally the breakdown was structured in multiple levels, the application was first broken down into two domains: One which concerns the handling of data within the software and another which concerns user interaction and control. The application was then further broken down as illustrated in Figure 3.4.

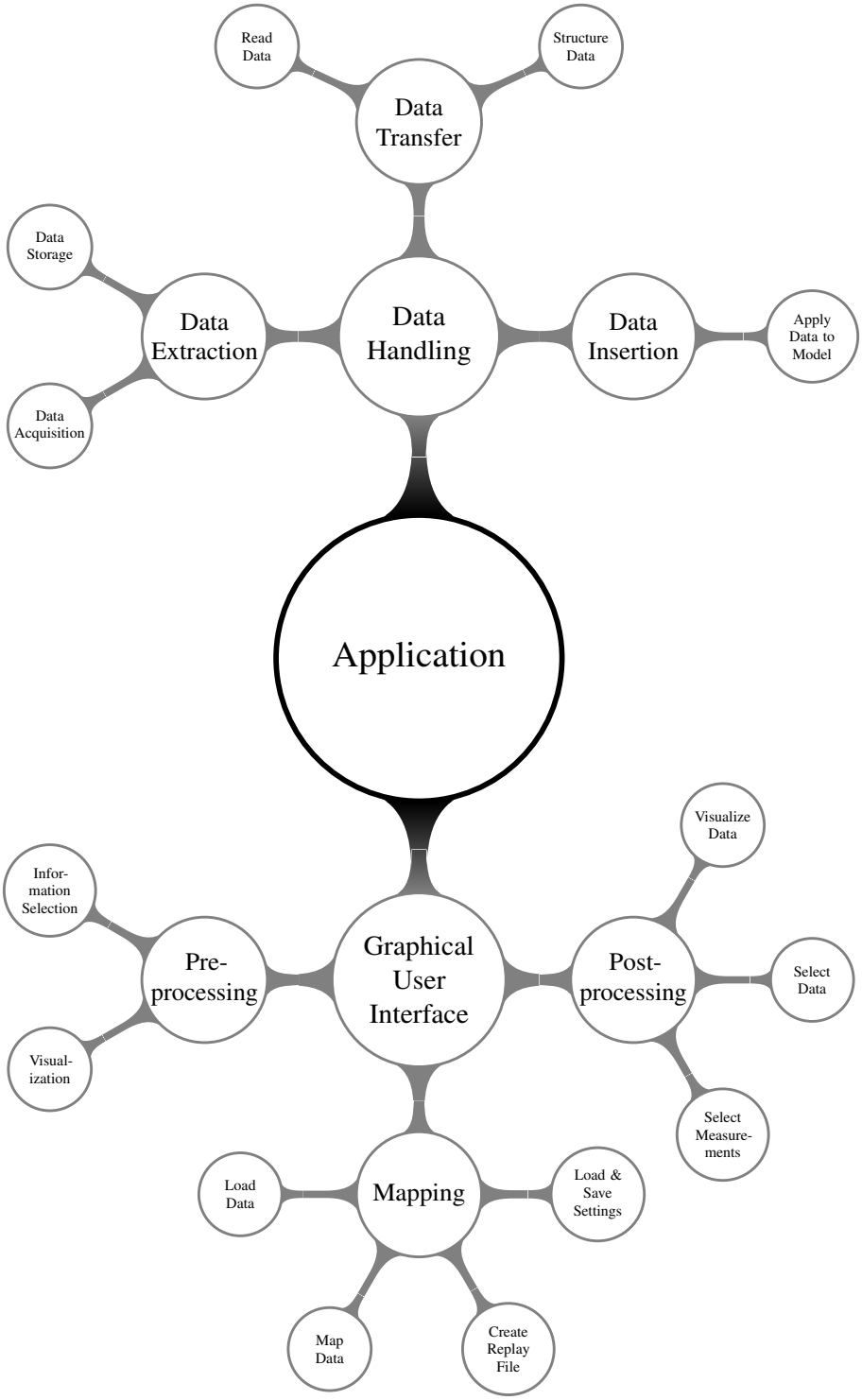


Figure 3.4: Illustrates the breakdown of the application.

3.4.1 Graphical User Interface

All interaction between the user and the application is handled through the Graphical User Interface (GUI). This meant that user friendliness and layout structure were of great importance throughout the development effort. The intention was that the GUI should help the user not only to map motions from ADAMS/Car to CATIA instances but also to preview motion data and assist in post-processing operations such as data visualization. These functions are handled through the three tabs; *Pre-processing*, *Mapping* and *Post-processing*.

In order to ensure a high degree of user friendliness a general structure, which all tabs were to be designed according to, was established. The general structure was set through discussions with the project supervisor who represented the design engineers of the Wheel Suspension department. Quite logically it was decided that the structure was to be based on that of a book written according to western customs, i.e. left to right, top to bottom. Further each step was to be enclosed in a numbered and labeled box. This may seem like an obvious way of structuring the layout but was nevertheless of great importance. An application with an ad-hoc GUI would add far less value to the customer and run the risk of being replaced or discarded quite quickly.

3.4.1.1 Mapping

In order to be able to use the motions extracted from ADAMS/Car to control the model in CATIA the motions must be connected to their respective parts in some way. However, as previously mentioned there is a mismatch in the data structures between the two software. The motions are exported from ADAMS/Car within the .res-file whose structure is explained in Section 2.5.3 and Appendix D, for generating replay files CATIA uses the structure explained in Section 2.4.3 and Appendix B. Thus the data must be re-structured to fit the structure used in CATIA. This is done in the *Mapping* tab of the application. In order to perform the mapping operation four functions were identified as necessary. The user must be able to read the product tree from CATIA, import the .res-file from ADAMS/Car, map motions to instances and create a replay file. As an optional function, to further enhance user friendliness, the ability to save and load a set of mappings was added. Figure 3.5 illustrates the complete work flow, with both key functions and optional functions.

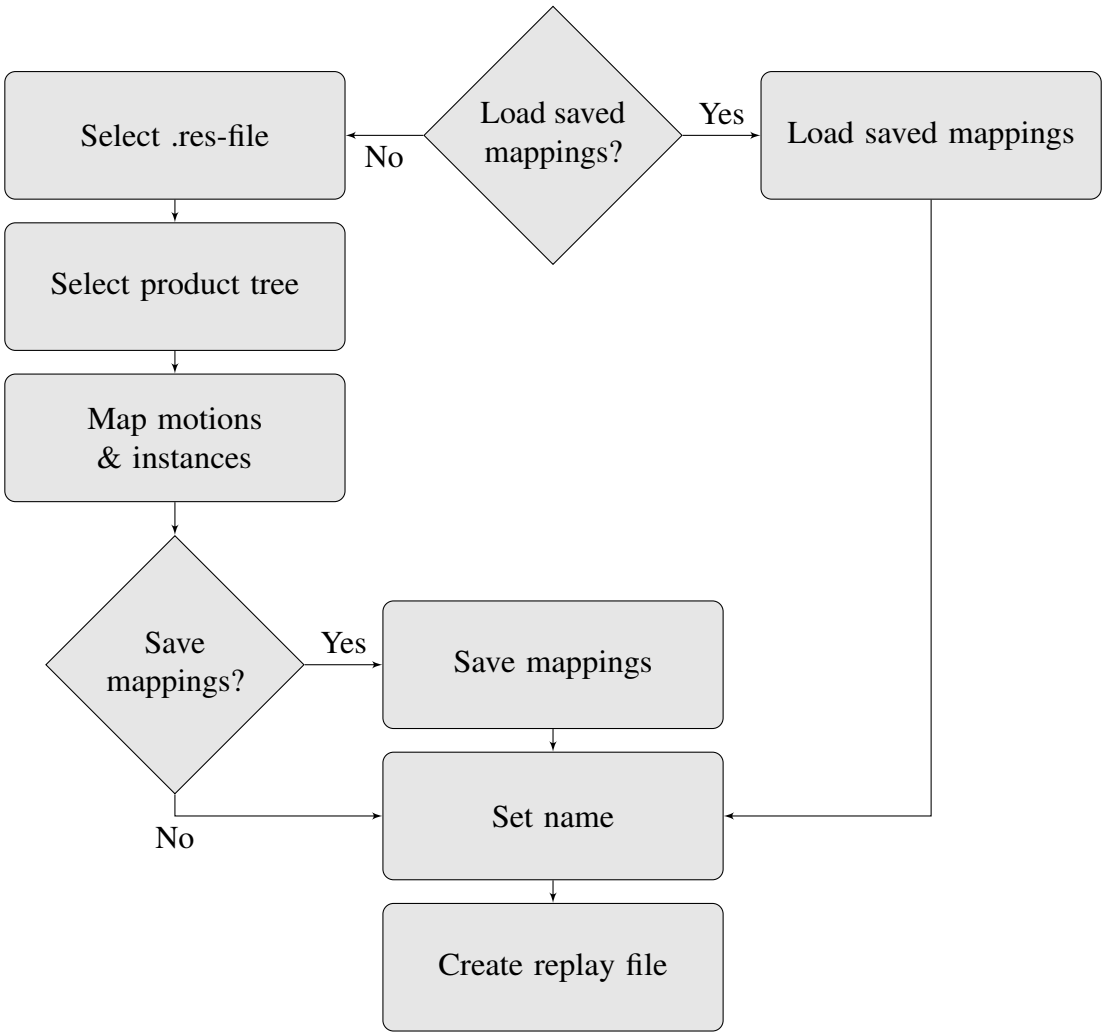


Figure 3.5: Illustrates the proposed process of mapping CATIA instances to corresponding motions from ADAMS/Car.

Figure 3.6 illustrates an early concept embodiment of the process illustrated in Figure 3.5. A step in the flow chart in Figure3.5 corresponds to a numbered box in the concept presented in Figure 3.6. The first step consisted of selecting and loading a .res-file from ADAMS/Car by browsing the windows explorer. Thereafter a product structure from CATIA was selected. The entire product structure, including all parts and products would be loaded. The instances and motions were, when loaded, visualized in the two curtain menus *Instances* and *Motions*. The mapping was carried out by selecting one instance, one motion and then pressing the *Map* button. When mapped the pair would appear in the **Mapped Items** box, but not disappear from their respective boxes. Once the desired components had been mapped the user would create a replay file by specifying and arbitrary name and pressing the *Create Replay File* button.

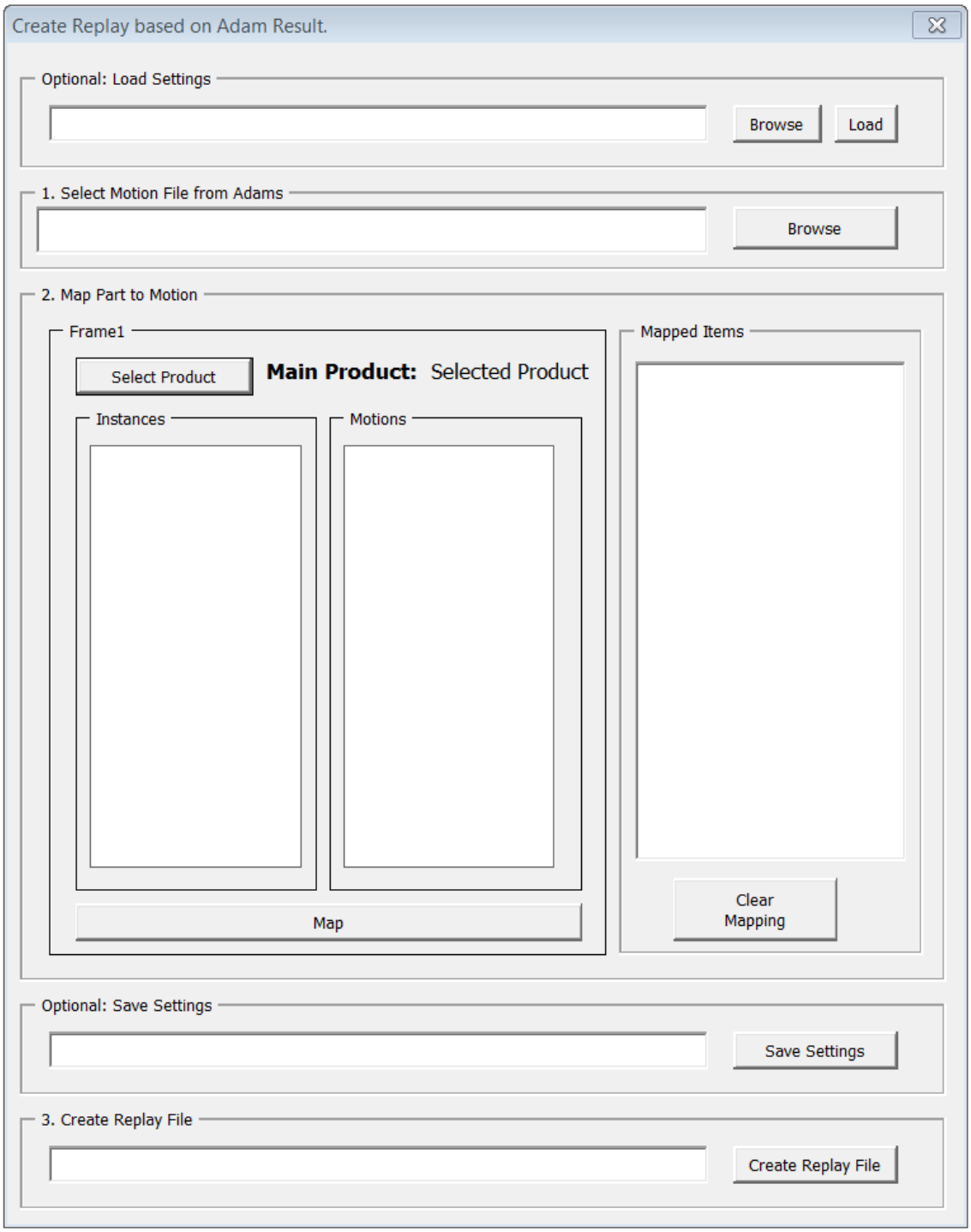


Figure 3.6: Illustrates an early concept of the mapping tab in the GUI.

Through testing and dialogues with future users it was soon realized that the mapping process was still time consuming, although not comparable with performing the task manually. As a product tree usually consists of a large number of instances looking up the right ones was time consuming. The same issue applied to the motions box, where only a

fraction of the motions were of interest. It was therefore decided that some kind of filtering function was needed in both cases. Moreover the mapping became quite "click intense" as three clicks were required for mapping one pair, therefore it was decided that the mapping button was to be removed. This would mean that a pair would be mapped as soon as one instance and one motion were selected, thus eliminating one click.

3.4.1.2 Pre-processing

During the mapping of instances and motions it could be of interest for the user to preview the motions in order to ensure that the correct data is selected. For this purpose the Pre-processing tab was developed, it allows the user to view a selected motion. Figure 3.7 illustrates the intended work flow. The .res-file which was previously loaded in the Mapping tab is transferred to the Pre-processing tab via a two-way link, this means that replacing the .res-file in the Pre-processing tab will also replace the .res-file in the Mapping tab. When the desired .res-file is loaded the motions can be screened in the same way as during the mapping process. Thereafter the desired motion is selected, the maximum and minimum value for each translation and rotation are illustrated, along with the timestep in which they occur. For further viewing there is to be a button which allows the user to export the motion to Excel where it can be plotted. The plotting was initially included in the GUI, however this solution was abandoned as VBScript does not support data visualization through plotting of a graph.

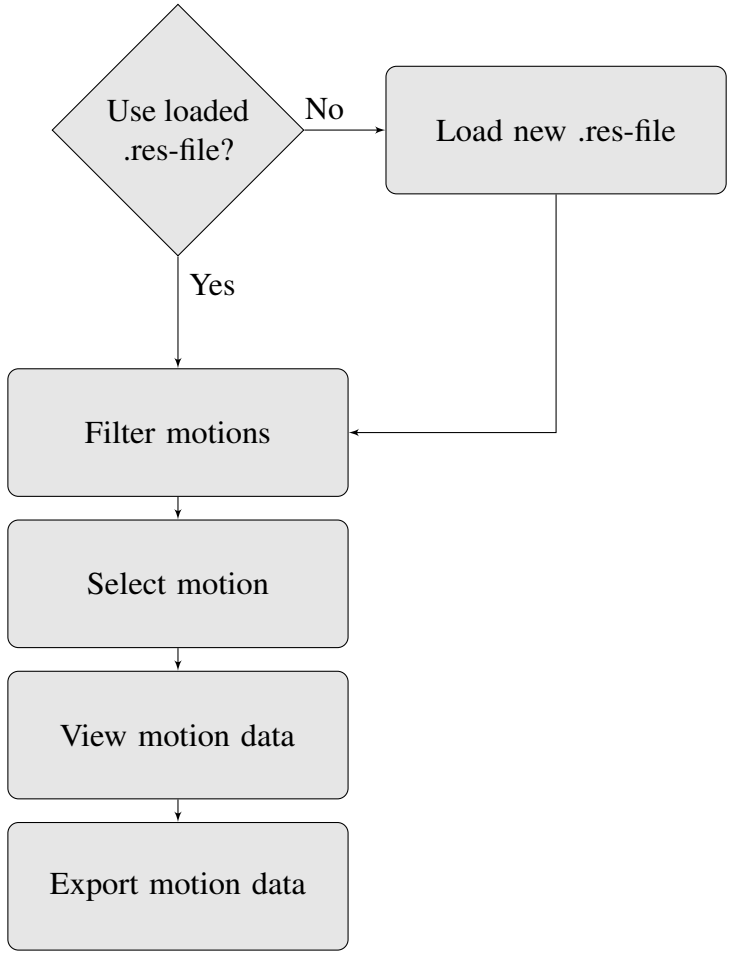


Figure 3.7: Illustrates the proposed workflow when using the pre-processing tab in the application.

3.4.1.3 Post-processing

The Post-Processing tab allows the user to perform analysis on a replay file. The type of analysis which is of interest here is clash analysis, i.e to measure certain distances between instances during a replay. The original concept for the Post-processor was that the user would simply select the replay and the application would then automatically set up suitable measurements. However, due to limitations of CATIA's ability to autonomously create and use measurements the Post-processor cannot be fully automated. Instead the user will need to set up the desired measurement manually. One benefit with this approach, compared to the fully automated one, is that the user gets a higher degree of freedom regarding the definition and use of measurements. The drawback obviously being increased lead time as the work must be done manually. However, since the engineers at the Wheel Suspension department already create measurements in their regular work the need for additional knowledge is not required.

Figure 3.8 illustrates the working process for the Post-processor. Before the tab can be used, the user must create all measurements which could be of use during the clash analysis. Once the measurements have been created, the post-processor be used to analyze replay

file. The user selects which replay file to be analyzed and imports all created measurements to the application. The user then selects which of the measurements he or she wants to include in the analysis. Once the replay file and the measurements have been selected the analysis can be performed. The application runs through the replay file and stores all measurement values for each timestep. After the analysis is complete, the user can view certain values directly in the application. The user can also export the data to an excel sheet for further analysis.

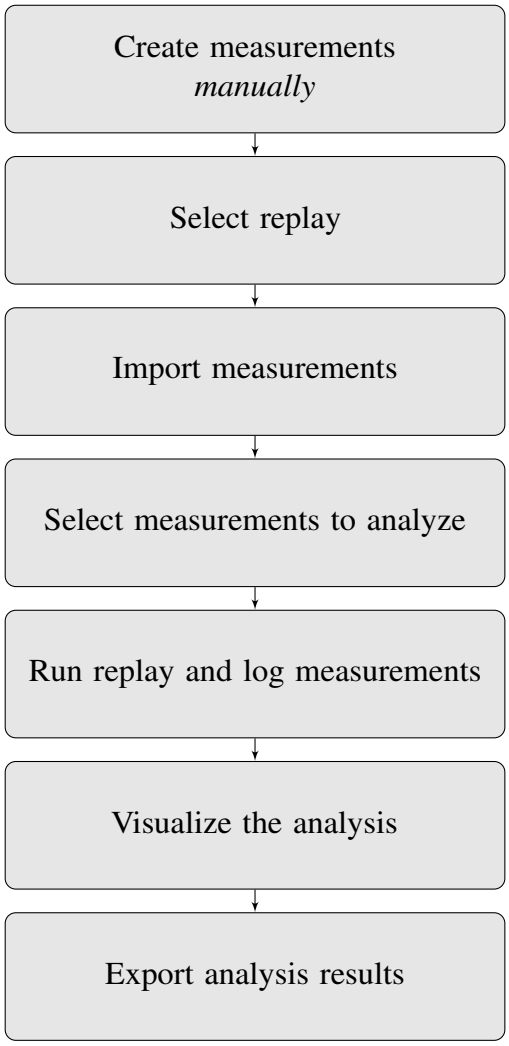


Figure 3.8: Illustrates the proposed workflow of the Post-processing tab. The first step is carried out manually outside of the application, while the rest of the steps are handled from within the GUI.

3.4.2 Data Handling

As the application will be based within CATIA, there are some basic functions which will be required. First the data from a drive event simulation must be extracted from ADAMS/Car. Second, the information needs to be transferred to, stored, and organized in CATIA so that it can be handled by the application. Third, the application needs a function which converts the data from the format used in ADAMS/Car to the format used in CATIA. These three functions will be described further in Sections 3.4.2.1 Data Extraction, 3.4.2.2 Data Transfer and 3.4.2.3 Replay File Generation.

3.4.2.1 Data Extraction

The first step in transferring data between ADAMS/Car and CATIA is to extract the desired data from ADAMS/Car. In order to do this, one must establish which rotation sequences and data structures the software use. As mentioned in Section 2.5 and further explained in Section 2.3 ADAMS/Car uses an Intrinsic 3-1-3 rotation sequence by default while CATIA works with an Extrinsic 1-2-3 sequence, explained in Section 2.3. Moreover the replay model is built in such a way that the body CS' are placed in the origin. This leads to a data mismatch between the two software. The motion of an arbitrary component relative to the origin cannot be used to control the replay model in CATIA. Using these motions would move the body CS along the same trajectory as the component moves. This would be fine as long as only translational movement is present, but as soon as rotations are introduced these motions become obsolete. The introduction of rotations lead to a mismatch as the body CS' are placed in the origin and not 4000-5000 mm away (as is the case for the rear wheel suspension in the x-direction). Consequently, the distance from the origin to the arbitrary component will work as a lever, distorting the components movement, as illustrated in Figure 3.9.

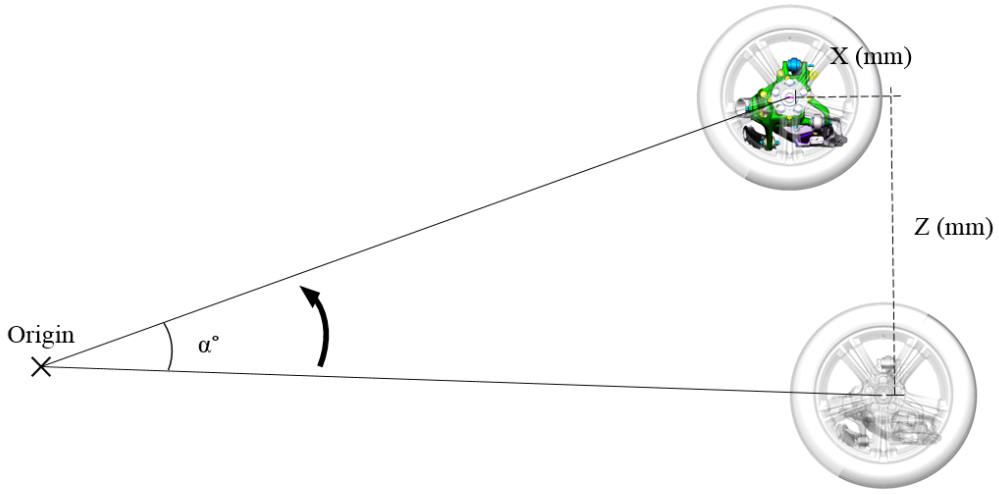


Figure 3.9: Illustrates how a rotational movement changes the position of the wheel suspension with a lever effect.

In order to extract motions which can be used for controlling the model in CATIA one must thus extract the motion of the arbitrary components thought body CS in ADAMS/Car. That is, logging the motion in ADAMS/Car such that is corresponds to the movement of the components body CS in CATIA. The issue here is that there is a mismatch between the placement of the body CS' in the two software, as body CS' in ADAMS/Car are not placed in the origin which is the case in CATIA. However, this is not the only issue at hand. One must also ensure that the extracted motions use the correct rotations sequence, which in this case is Extrinsic 1-2-3 rotations.

The conventional way of logging and extracting motions from ADAMS/Car is through placing a marker on a node of the component of interest and then specifying a request for the motion of the marker, this process is described further in Section 2.5. By doing so the motion of the specific component, relative to the origin is obtained. As mentioned these motions are not of any use due to the lever effect. However, by modifying the marker such that it is placed in the origin and connecting it to the component of interest with a rigid axis the lever effect can be taken advantage of. This means that the motion obtained is that of the thought component body CS, as it is placed in CATIA. As the model is kinematic, an arbitrary node on the component of interest can be used to log the motion. In order to implement this way of logging motions Vehicle Dynamics must update all affected templates with the modified markers, this is however a one time task and not repetitive work.

Even-though the correct motions can now be logged they must still be transformed to the correct rotation sequence. When specifying a request ADAMS/Car offers two standardized ways of exporting rotations. These are as Intrinsic 3-1-3 rotations or Extrinsic projected 1-2-3 coordinates [31]. However, through writing a script which is called by the request motions can be extracted on any of the 24 available rotation sequences. Once this feature was discovered the necessary script was developed in collaboration with MSC Software. The script which was written in C is called upon by the request, thereafter motions are transformed from Intrinsic 3-1-3 rotations to Extrinsic 1-2-3 rotations and stored in the .res-file in this format. The script was written in C partly because pre-defined conversion functions could be called from ADAMS/Car and partly due to the ease of implementation and maintenance. Volvo Cars already use similar scripts in other areas of its development work, consequently the company already has employees who are responsible for such scripts as well as defined processes for how to implement and maintain such scripts. The script would be compiled and maintained in a central database together with other similar scripts. Now that the correct motions are logged and stored in the .res-file the next step is to transfer the data from there to CATIA.

3.4.2.2 Data Transfer

Since the application is developed in CATIA the data must be transferred from ADAMS/-Car's .res-file into CATIA, in order to be stored and re-organized. However, CATIA does not currently support the .res-file format. One way to solve this problem is to let CATIA read the .res-file as .txt file, by doing so each line will be handled as a separate string. The requested data can then be extracted from the strings. This solution has both pros and cons for the functionality of the application.

One advantage with extracting data line by line is that the application will be robust with respect to file sizes and what types of information the .res-file contains, as long as the general structure is the same. Another advantage is that adding more functionality for extracting other types of data will be simple. The only thing that needs to be added to extract additional data is a case that defines what happens to this additional data. Reading the file line by line also enables other engineers, without deep programming knowledge, to study and understand what is happening in the code.

There are however some disadvantages to reading the file as a .txt file, one of these is the sensitivity to structural changes in the .res-file. As all the extracted data is dependent on the string of information and the declaration structure which the .res-file uses. If the declaration structure is changed, e.g. through a patch, the application may no longer be able to read the .res-file and extract the requested data. Thus, if the declaration structure is changed the application will need to be updated to be compatible with the new declaration structure. Depending on the extent of the changes in the .res-file the application may only need an update of certain indices and names, however if the whole structure of the .res-file is changed it is likely that the application will need to be re-written completely.

Another disadvantage is that this method is relatively slow compared to other methods for data extraction from a file. But for this application, that speed reduction will have minimum effect for the user. It would be more efficient to optimize the code than to use another extraction technique. One final disadvantage, which can also be an advantage depending on whose perspective to take, is that by the selected method for extracting data is bi-stable, i.e. it either works or it does not. This is an advantage from a maintenance perspective as it becomes clear when changes need to be made. it is also an advantage in the sense that the user will not be able to extract the "wrong" information. The pros and cons of extracting data by reading each line as a string are summarized below.

+ Robust against file size variation.	− Dependent on name structure.
+ Simple to add additional functions.	− Relative slow process.
+ Easy to study.	− Difficult to update to new structure.
+ Either it works or it doesn’t.	− Either it works or it doesn’t.

The process of reading and extracting data from the .res-file can be divided into the three main steps which are illustrated in Figure 3.10. In the first step it is declared how the extracted data will be stored and organized. The declaration also defines which data will be extracted based on the mapping that the user has declared, the mapping process is explained in Section 3.4.1.1. The second step is to obtain the IDs for all requested motion values. This is done so that the application knows where in the .res-file the motion values are located, for a description of the .res-file structure see Section 2.5.3. The third and final step is to extract the actual motions values from the .res-file and store them at the declared location within the application. The remaining text in this section explains the three steps illustrated in Figure 3.10 in greater detail.

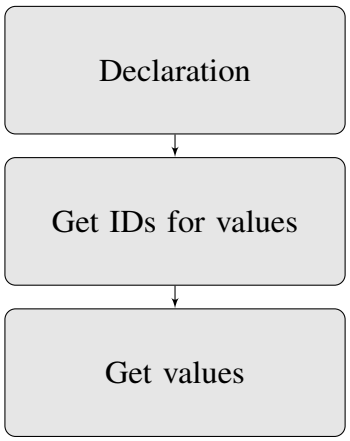


Figure 3.10: Illustrates the general process for transferring data.

The first step in the main process is to declare where and how the extracted data will be stored and how it will be organized. The first step in the declaration process is for the application to receive a list stating which data to extract. The list has been defined by the user when mapping instances to motions, see Section 3.4.1.1. Each item in the list contains one instance, from CATIA, and one motion, from ADAMS/Car, which will be connected. The instance and its motion are extracted, for every item, and then used to create a class which can hold all the extracted data from the .res-file. After all items have been read and their corresponding classes been created the ID and other general information from the .res-file can be extracted, Figure 3.11 illustrates this sub process.

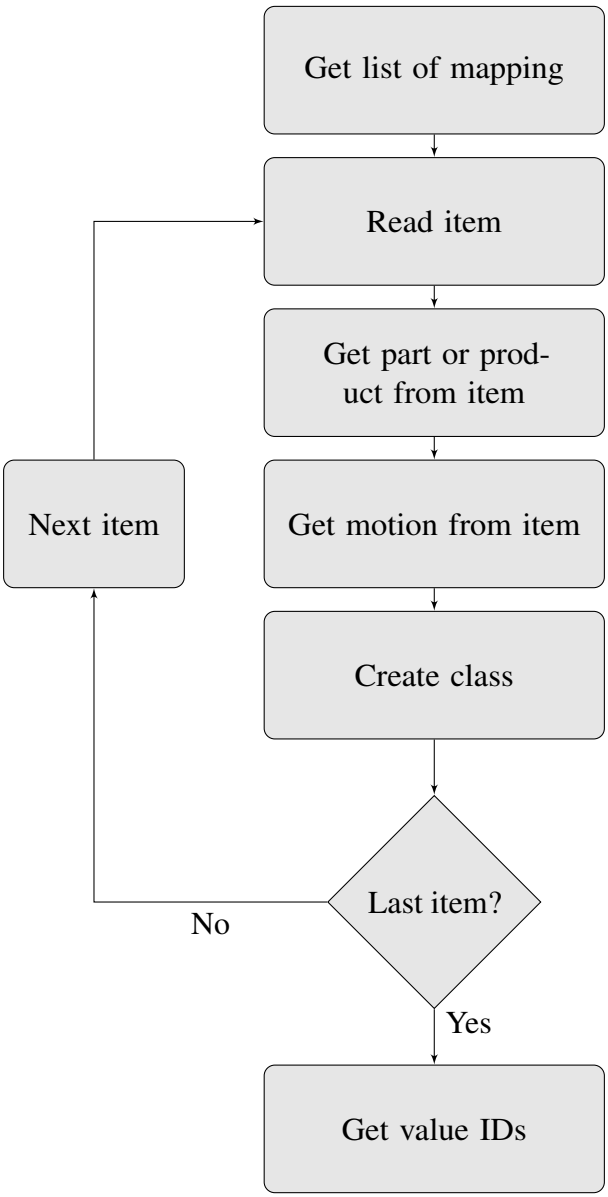


Figure 3.11: Illustrates the first step in the data transfer process. Where and how the extracted data will be stored.

The extraction of the IDs and other general information is based on reading each line of the file as a string of information. The application loops through what essentially is an *IF-statement* for each line in the .res-file. When one line has been read, the code first checks if this *is a valid line to extract data from?*. Weather this statement is true of false depends on which motions the user has mapped. Apart from the validity check of the line, the application also checks what kind of information can be extracted. There are two types of information that can be extracted. Case A is that the motion name, object type and motion ID are extracted. Case B is that the ID for a translational X, Y, Z or rotational Psi, Theta and Phi value is extracted. When all lines in the .res-file have been read the .res-file will be read once more in order to extract the specific values for all IDs. The process for extracting the IDs and general information is illustrated in Figure 3.12.

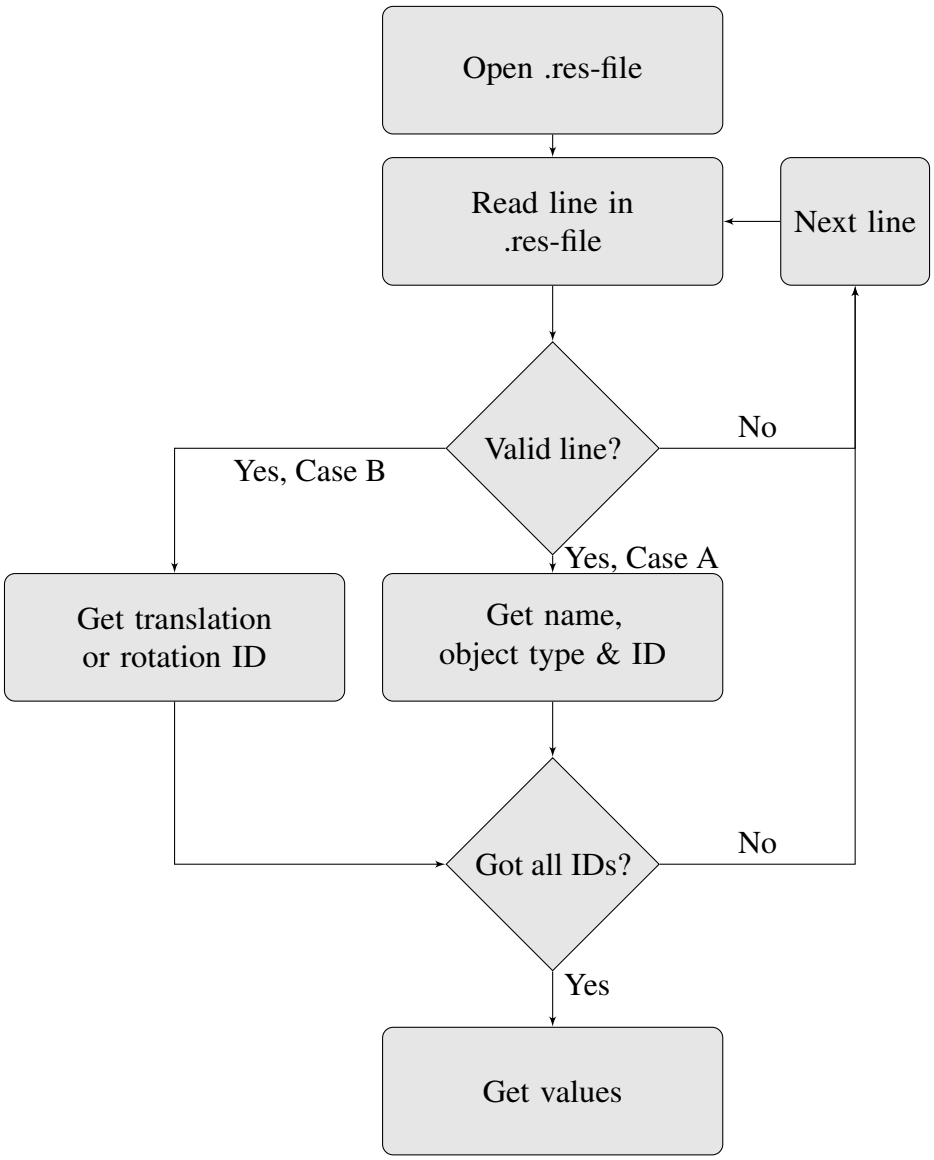


Figure 3.12: Illustrates the second step in the data transfer process. How the ID (position) of each value in the .res-file is located.

The final step in the main data transfer process, illustrated in Figure 3.10, is to obtain the values for each ID. In order to do this the .res-file is reopened and read again. The same approach is used as when the IDs were extracted, the application loops through the .res-file with an *IF-statement* to identify which lines are valid to extract data from. If the line contains any data which is of interest this data is extracted and stored under the location specified in step one of the main process. Once all requested values have been extracted the data transfer process is complete. Now the data can be used to generate a replay file, this process is explained in the next section, Section 3.4.2.3.

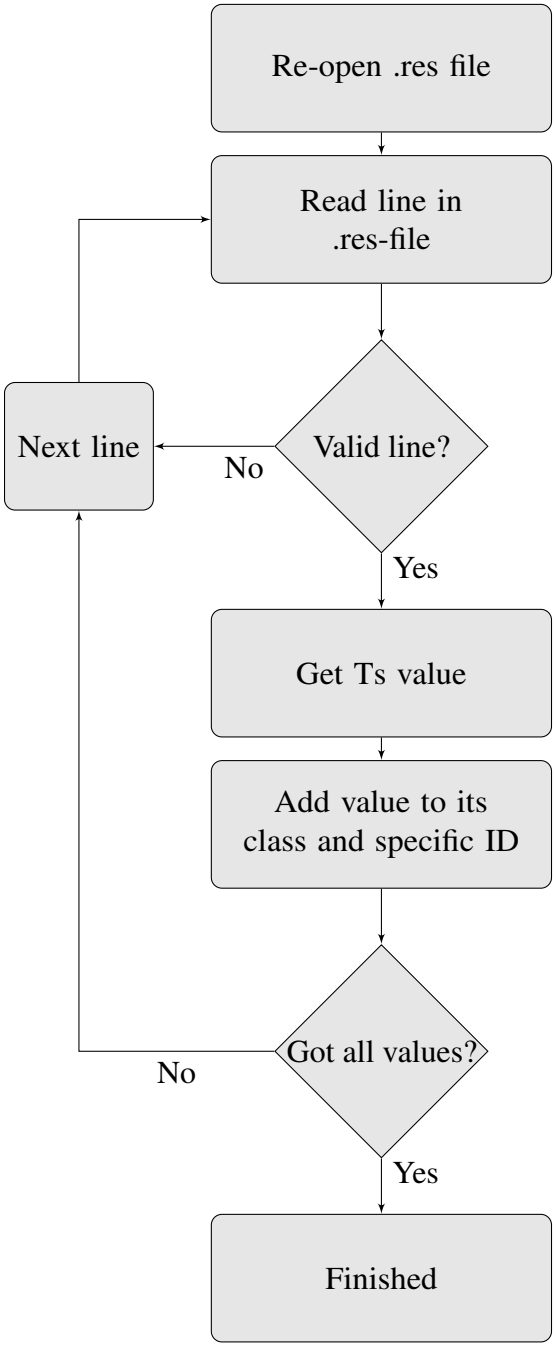


Figure 3.13: Illustrates the third step in the data transfer process. How the values are obtained and stored.

3.4.2.3 Replay File Generation

There are two main ways to generate a replay file. The first way is to through setting up a mechanism, see Section 2.4.2 for an explanation of this approach. The second way is to generate the replay directly through CATIA's API. Both approaches have their pros and cons, due to high degree of automation requested in this project the replay will be generated through CATIA's API. If the replay would be generated from a mechanism, it would require that the user defines all joints and connections between the instances. This in

turn would require more time from the user when working or longer development time of the application due to the added complexity. It would also mean that transferring motions from ADAMS/Car would not be possible.

As mentioned in Section 2.4.3 the replay file generation through CATIA's API consists of two main parts. First the user must define which instances are to be included in the replay file. Second, the translational and rotational motions for each timestep must be added. Translational motions are easy to add as they require no conversion. However, rotations are a more complex matter. The rotations extracted from ADAMS/Car are extracted as Extrinsic 1-2-3 Euler angles, while CATIA uses Extrinsic 1-2-3 rotation matrices to define and handle rotations. Thus a conversion from Euler angles to rotation matrix format is necessary. Note that it is important to select the correct conversion formula as there exist 24 different ones, one for each way of defining rotations.

The process for generating the replay file is illustrated in Figure 3.14. The first step is to take the data which has been organized and stored from the data transfer and create corresponding rotation matrices for each instance and timestep. The matrices are calculated according to equation 3.1, where ψ, θ and θ corresponds to the projected rotations around the X, Y and Z axis. Each element of the matrix together with the translations are rearranged into a single vector formatted as illustrated equation 3.2, which is the format that CATIA uses.

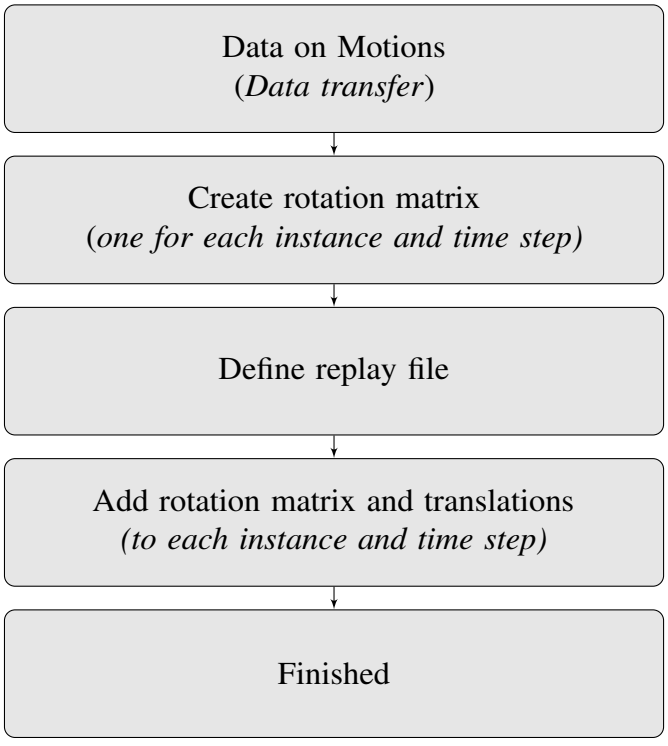


Figure 3.14: Illustrates the process of how a replay file is generated in CATIA.

$$R_{123}(\phi,\theta,\psi) = \begin{bmatrix} c_\theta c_\psi & c_\theta s_\psi & -s_\theta \\ s_\phi s_\theta c_\psi - c_\phi s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\theta s_\psi \\ c_\phi s_\theta c_\psi + s_\psi s_\psi & c_\phi s_\theta s_\psi + s_\phi s_\psi & c_\theta c_\psi \end{bmatrix} \tag{3.1}$$

$$ArrayOfVariantOfDouble = \begin{pmatrix} R_{11} \\ R_{12} \\ R_{13} \\ R_{21} \\ R_{22} \\ R_{23} \\ R_{31} \\ R_{32} \\ R_{33} \\ x \\ y \\ z \end{pmatrix} \tag{3.2}$$

Once all rotation matrices has been created the replay file has been defined. In other words, the instances which are to be included in the replay file have now been added. The user specified which instances to include when mapping the instances to motions. The last step is to add all rotational and translational motions to its respective instances for each timestep.

3.5 Detail Design

The purpose of the detail design phase was to identify errors and bugs in the code, gain user input on the layout and functionality as well as gaining an understanding of additional functionality from the users perspective. In order to achieve this the phase was initiated with a pilot test run where three design engineers for the Wheel Suspension department tested the application.

When the pilot study was conducted, the whole application could not be tested. The state of the application was that the *Mapping* and *Pre-Processing* tabs were fully functional. While the *Post-Processing* tab was only the graphically implemented, i.e. no functions were included in this tab. The pilot run was initiated with a common meeting where a short presentation, regarding how to operate the application, was held. This material was thereafter distributed and the application set up on each computer. Three different .res-files were distributed as well, one containing a fully kinematic run, one where bushing stiffness was included and one which was fully compliant, i.e. bushing stiffness and flex bodies were included. Thereafter the design engineers were given one week to explore the application and its functions. The pilot run was closed with a common meeting where feedback was gathered. before closing the meeting a priority list was set up in collaboration with the participating design engineers. Quick fixes were listed as first priority while added functionality was listed as second priority and ranked internally.

There was plenty of feedback received, both regarding errors and bugs but also regarding additional functionality. The general feedback regarding errors and bugs was that much of this had already been handled. There were however some fixes to be done, for example the application crashed when the user pressed *"Save Current Settings"* in the mapping tab and then pressed the red cross in the dialogue box which appears. Another example is when the user loads a saved mapping in the *Mapping* tab and then wants to preview the information in the *Pre-processing* tab. Normally the .res-file is loaded in the tab as well as they are linked, however the link had not been implemented for loaded settings, only for selection of a new .res-file. There were also comments from multiple users that the depth selection function when loading the product tree was unintuitive and overly complicated. Several suggestions on how to fix this were gathered and through a discussion is was concluded that the current design with two buttons, *Add Products to List* and *Add Specific Instance to List*, and one box for typing the depth using integers was to be replaced. The issue with the current layout was the depth selection box, where the user could type the word *All*, or any given number, then select an instance and load either all levels below the selected instance or a set number of levels below. This was unintuitive because the user may not know the number of levels, the models usually don't have that many levels and using the *All* command did not load all instances in the product tree. Instead a curtain menu illustrating the number of levels in the product tree was discussed, as was arrows pointing up and down for selecting depth. The problem with the two solutions is that logging the number of levels requires a product tree to be selected first. Instead a two button layout was agreed upon. One named *Add All Instances Below* and another named *Add Specific Instance*, even-though the buttons are quite self-explanatory the first one adds all instances below the selected one while the second on adds only the selected instance. Other quick fixes included adding explanations to the abbreviations used in ADAMS/Car naming convention, i.e. FBL - Flexible Body Left-side and GEL - General Part Left-side.

Regarding additional functionality there were multiple suggestions, one which was met with great enthusiasm, both from the authors and the other design engineers was the addition of a tab which contained a table with a set of parameters which are often used to verify that the simulation run is adequate. This was the thought functionality of the Pre-processing tab, however using parameters requested by the future customers and users would be a better way of doing this. The suggested layout of the table is illustrated in Table 3.1 on the following page.

All the requested data is available, either by using current markers and requests or by creating new ones. The data would be inserted in the table automatically when a .res-file is selected in the *Mapping* tab. The box *Curb + Z* will be a text box where a distance, e.g. 10 mm, from the Design Position (DP) in ADAMS/Car can be entered in the form of an integer. This will update the remaining text boxes in the row with each parameters value when the wheel suspension is in the specified position. This table will be implemented in its own tab for now, However in the future it is likely that it will replace the *Pre-processor*. The reason for not doing this right away is to not remove any functionality until further tests have been done, in case the *Pre-processor* proves to be useful for other contexts.

Table 3.1: Illustrates the purposed layout of the "simulation verification" table, from the pilot study.

	Rebound	Jounce	ADAMS' DP	Curb + Z
WLC Z Coord				
WLC Z Travel				
Toe Angle				
Camber Angle				
Damper Travel				
Damper Length				
Rack Travel				

Another additional functionality which all participating design engineers requested was the ability to generate shorter replay files between certain timesteps. This is important as a clash analysis may only concern a small number of timesteps, making it redundant to include all timesteps in the replay file. This function would thus save the design engineers time, while simplifying their work. The feature will be implemented by adding a Section to the *Mapping* tab where two textboxes, one for specifying the first timestep and one for specifying the last, will be included along with a *Create Replay* button. The replay file will be named using the same textbox as when creating a replay file including all timesteps.

Feedback on the layout was positive as the designed workflow, with numbered sections from top-left to bottom-right, was intuitive to understand and follow. A discussion regrading the naming convention of markers and requests was also initiated and will be continued by involving Vehicle Dynamics.

4
Results

This chapter presents the results of the master thesis project, beginning with the new proposed working methodology for performing a packaging analysis of critical components. Thereafter there is a section on ADAMS/Car, followed by one about the Application.

4.1 New Working Process

One of the main targets of this master thesis project, see Section 1.2, was to automate the working methodology for creating design spaces of components in the wheel suspension. The approach used for this was to develop an application which automates certain steps in the working process. By doing so a new working process has been developed, this process is illustrated in Figure 4.1 below.

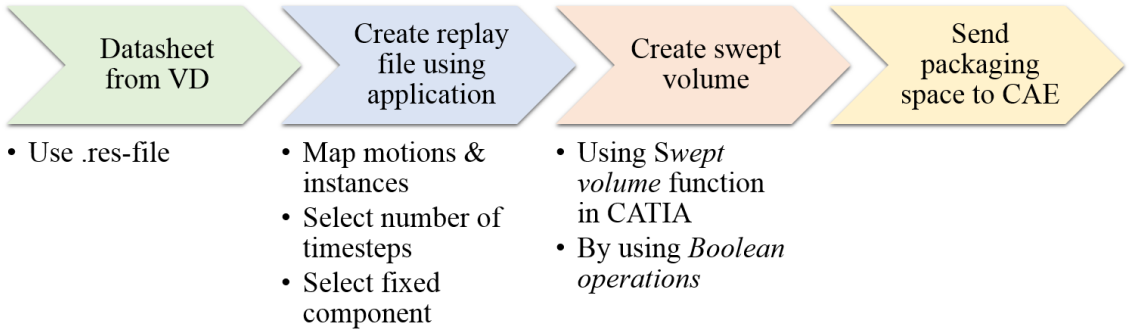


Figure 4.1: Illustrates proposed working process which is enabled by using the application developed during this master thesis.

In the first step of the new working process the design engineer receives a .res-file for the requested drive-case from Vehicle Dynamics, thereafter the application is used to map the motions to instances in CATIA and generate a replay file, as described in Section 4.3.1. Thereafter the generated replay file can be used to create design spaces by fixing one component and moving the other components relative to it. The design space is, in this case, the volume which has not been intersected by the other components. The design space can then sent to CAE for topology optimization prior to design realization.

In order to compare the current working process with the one developed in this master thesis project the steps which correspond to each other were identified and are illustrated in Figure 4.2 through a common coloring scheme. The main difference being that, through the

application, the new process replaces three steps in the old process with only one. However, starting from the beginning the first step, marked in green, is to receive input data from Vehicle Dynamics. The is currently done in the form of a vast datasheet in excel, meaning that Vehicle Dynamics invest time in preparing this datasheet. The process developed in this project uses the .res-file which is automatically generated for each simulation, meaning that no extra post-processing work is necessary for Vehicle Dynamics.

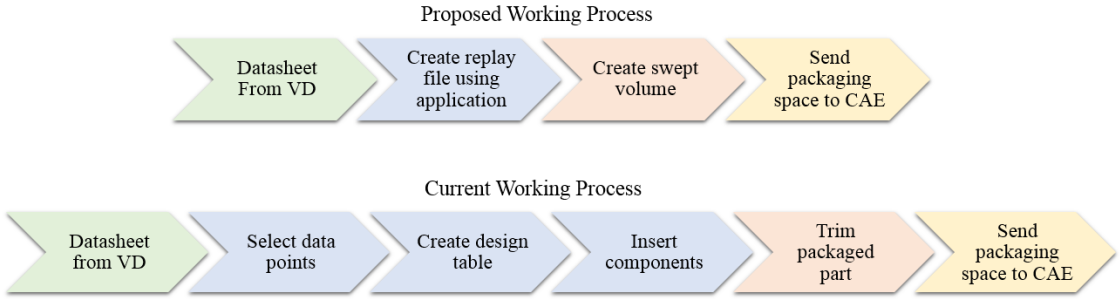


Figure 4.2: Illustrates the two working processes, where corresponding steps are colored in the same color.

The biggest improvement however involves the next three steps of the current working procedure, which are described in detail in Section 3.1. These cumbersome steps have, been reduced to one step which is carried out in a fraction of the time, through automation of the data transfer process. The involved design engineers estimate time savings of roughly 50 effective working hours for packaging of a critical component. Further the precision of the analysis has also been increased, partly due to the ability to handle all timesteps instead of a set of measuring points, this is illustrated in Figure 4.3 by a basic example, where it is clear that the number of measuring points increases the accuracy of the motion. The increase in precision is also due to the ability to handle new data. By transferring the motions, instead of mimicking them, bushing stiffness can also be accounted for. The generic rules used can thus be verified and replaced with the actual motions transferred from ADAMS/Car.

The steps colored red partly correspond to each other. In the current working process *Insert components*, which is colored blue, includes inserting the components at each of their five design positions and the step named *Trim packaged part* corresponds to subtracting the other components from the one to be packaged by using Boolean operators. In the new working procedure the volumes which are to be subtracted from the part which is to be packaged are created using CATIA's *Create swept volume* function, which takes minutes to perform instead of hours. Thereafter the design space is created by in the same way as in the current working procedure, by using Boolean operators. It is still necessary to perform one packaging analysis and create design spaces for each powertrain setup, this was however necessary in the current working procedure as well.

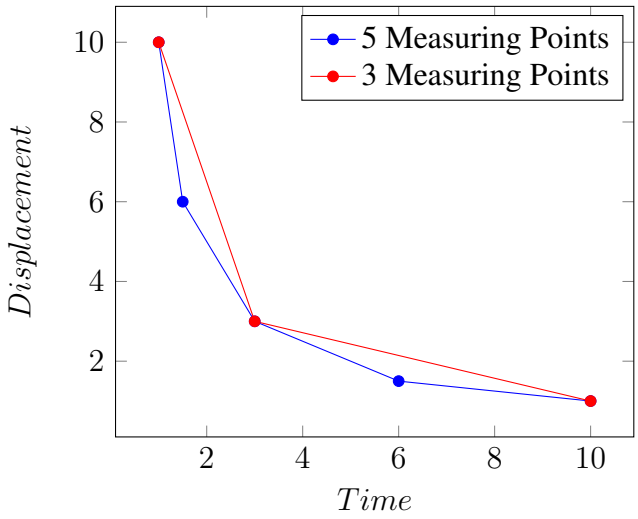


Figure 4.3: Illustrates how increasing the measuring points increases the accuracy of the motion.

As a final step, which is unchanged in both processes the generated design space is sent to CAE for topology optimization. However, the design space produced using the new working process has a higher level of detail.

4.2 ADAMS/Car Template Modification

In order for the application to function there are some changes which must be made in ADAMS/Car. In order to extract motions with the correct rotation sequence customized markers and requests must be placed, as described in Section 3.4.2.1. As the current setup is designed to work only with rigid bodies each body needs only one marker and request which monitors the motion of an arbitrary node on the body. As customized requests must be added a naming convention was set during the project: *annotation_body.type_instance.name_hardpoint*. For the knuckle the request would be named as follows: *wsa_fbl_KNU_pt9*, where wsa is an abbreviation for wheel suspension application, fbl for flexible body, KNU for knuckle and pt9 for hardpoint 9. Note that this naming convention was merely set for use during the project and includes the information which was believed to be important for the user. A long term naming convention is under development as discussions were intiated during the pilot study. Where it was agreed that some type of prefix (right now wsa) must be included, as must the instance name and hardpoint.

4.3 Application

The final version of the application will be presented in following sections. However, as this application is Volvo Cars property only the GUI will be presented. The descriptions regarding routines and the code are summarized in a separate document.

4.3.1 Mapping

The main feature of this application is the *Mapping* tab which allows a user to create an replay file based on a .res-file from ADAMS/Car. Figure 4.4 illustrates the *Mapping* tab in the final version of the GUI. The working process of the mapping procedure is illustrated in Figure 4.5.

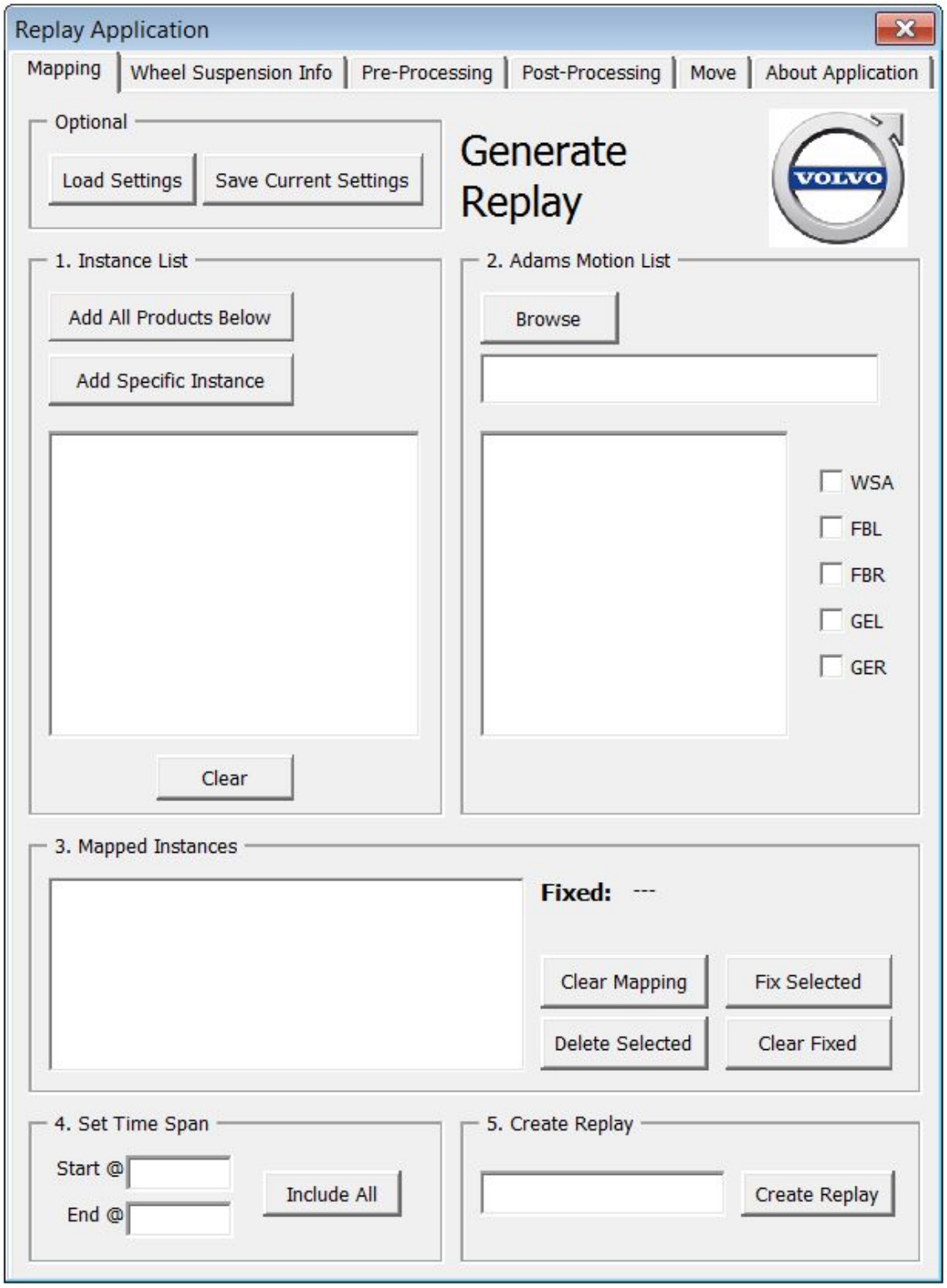


Figure 4.4: Illustrates the final version of the mapping tab in the GUI.

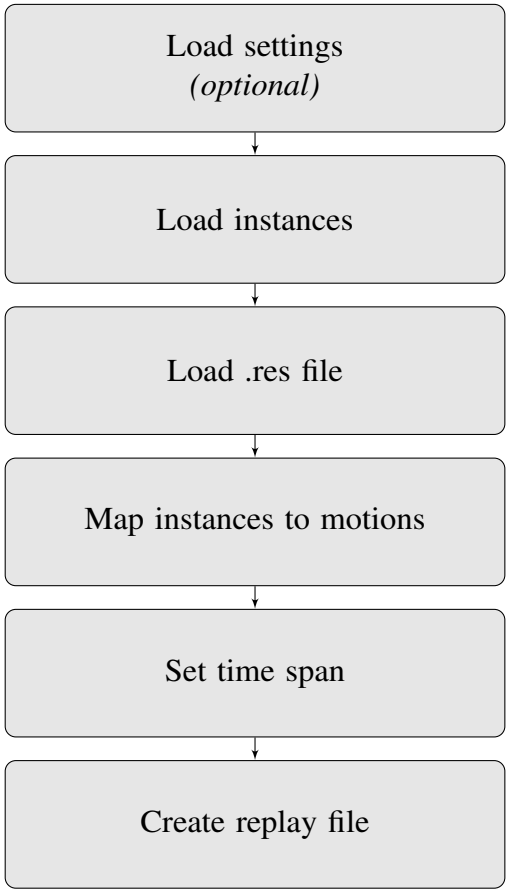


Figure 4.5: Illustrates the proposed working process for the final version of the mapping tab in the GUI.

The user begins by loading the desired instances from CATIA, this i done in frame *1. Add Instances*. There are two ways of loading instances, the user can either use the button *Add all instances below* or *Add specific instance*. The former of the two allows the user to select one instance from the product tree in CATIA, loading all instances below the selected one. The later of the two allows the user to add a specific instance to the text box. When an instance is loaded it will appear in the textbox of the application, the *Clear* button allows the user to clear the text box.

The next step is load data from the .res-file, this is done in the frame *2. ADAMS Motion List*. By pressing the *Browse* button the user can select a file stored somewhere on the computer or a server. The only requirement is that the computer which is used can access the file through a file path. Once a .res-file has been selected and loaded the textbox below the *Browse* button will contain all motions which are included in the file. By using the filter boxes next to the textbox the user can filer out motions so that only the relevant ones remain.

The third step is to map the instances and the motions together. This is done by selecting one instance from the textbox in the *1. Add Instances* frame and selecting one motion from the textbox in the *2. ADAMS Motion List* frame. The application will then automatically

map the instances, as described in Section 3.4.1.1, which will appear in the textbox in the *3. Mapped Instances* frame. If the user would like to redo the mapping he or she can either clear all mapped pairs by clicking the *Clear* button, or remove a specific mapping by using the *Delete Selected* button. The button *Fixate* allows the user to change the reference from the global CS to that of a specific instance, by doing so a replay file which shows the movement relative to the selected instance can be generated. The fixed instance can be un-fixed by using the *Clear fixed* button, which sets the reference back to the global CS again.

Before the replay file can be created, the user must set which time span the replay shall cover, this is done in the frame *4. Set Time Span*. The default is be to include all timesteps from the .res-file. The user can for instance create an replay file which just includes timesteps 10 to 20 by entering these values. The button *Include All* allows the user to set the time span to the default value and include all time steps.

In the fifth and final step, *5. Create Replay*, the user can set a name for the replay file. If no name is set, the replay file will be assigned a default name. Thereafter the user can press the *Create Replay* button and the replay file is generated.

An optional feature of the *Mapping* tab is that the user can save or load a settings file containing a set of mappings. The settings file allows the user to store a certain mapping configuration for re-use. Note that in order for the settings file to work the file path which leads to the .res-file must still be valid. It is also possible to load a settings file and then load a new .res-file, while keeping the mapped motions and instances.

Regarding the level of error handling the application will point out if something is wrong and give a suggestion to what the issue could be, the application does however not correct any mistakes. For instance, if one instance has been mapped with multiple motions the application will simply point this out and not allow the user to generate a replay file. To be able to generate the replay file the user must first fix this issue on his or her own.

4.3.2 Pre-Processing

The Pre-processing tab allows the user to examine the data contained in the .res-file and export the data for further analysis, if desired. Figure 4.6 illustrates the final version of the *Pre-processor* tab.

In the frame *1. Select Motion* the user can select a motion by using the *Load* button. Once the motion has been loaded basic information about the motion is shown in the frame *Basic information*. The information shown is as follows; min and max values for the X-, Y-, Z-translations, as well as X-, Y- and Z-rotations, along with the timestep at which these values occur. All motion data comes from the .res-file which is loaded in the *Mapping* tab. If the user wants to analyze a motion in greater detail he or she can export the data to an excel sheet by using the *Export* button. Next to the textbox containing the motions, are a set of check-boxes which the user can use to filter the motions so that only the relevant ones are shown.

Similar to the *Mapping* tab, the error handling only points out if there is an issue. In this case the user will receive a warning message and wont be able to proceed with the Pre-processing operations. To be able to continue the user must first attend to the issue at hand.

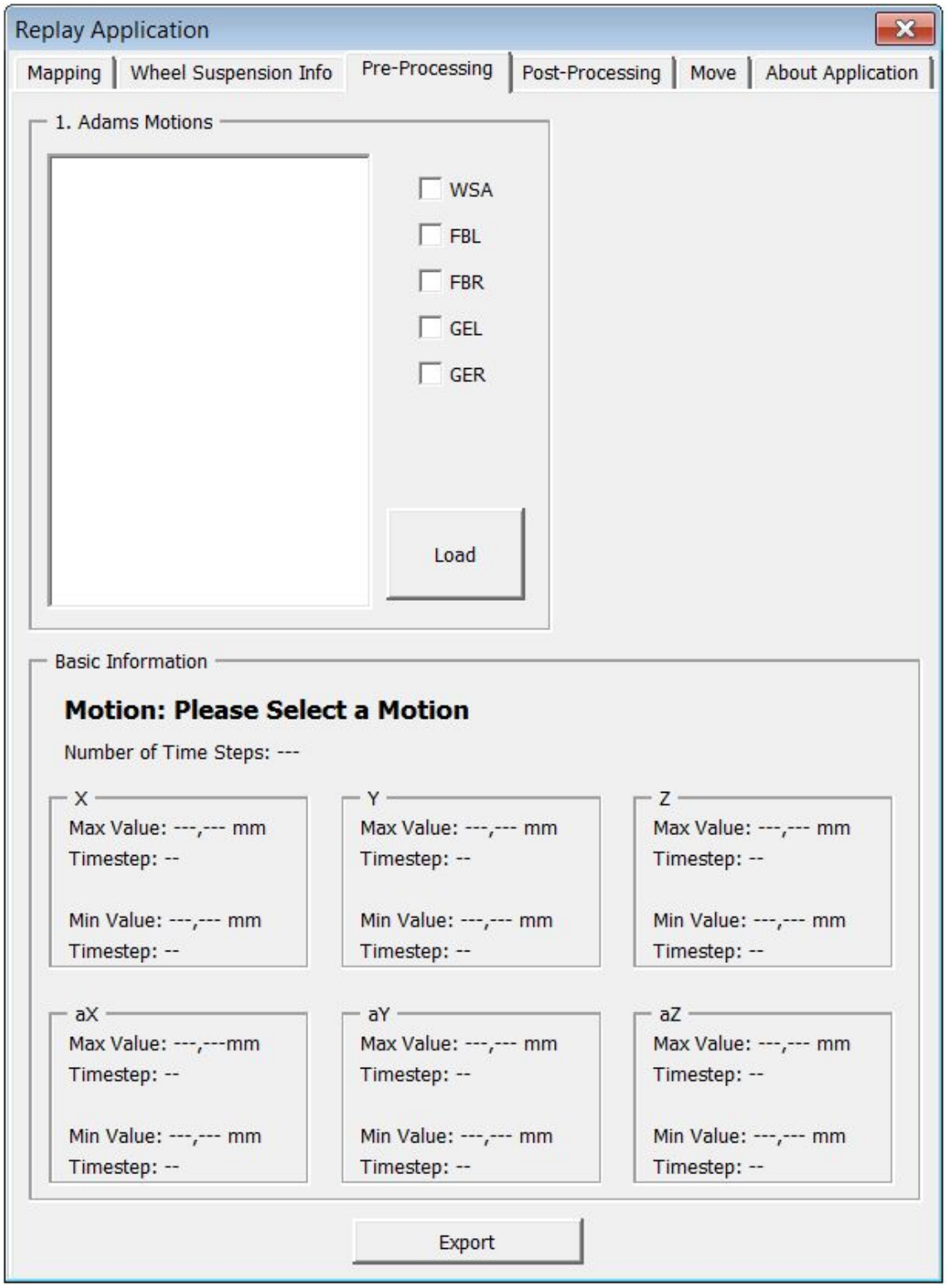


Figure 4.6: Illustrates the final version of the Pre-processing tab in the GUI.

4.3.3 Post-Processing

The Post-Processor allows the user to study and analyze measurements during a replay. Before the post processor can be used, the user must manually create the measurements in CATIA's main product. In order for the Post-Processor to work at least one measurement must be placed. The layout of the *Post-processing* tab is illustrated in Figure 4.7.

Once the user has placed the desired measurements the post-processor can be used. The first step is to select a replay file from the product tree, by using the button *Select Replay*. Thereafter the measurements which the user has created can be imported into the application. All imported measurements are shown in the textbox above the *Import Measurements* button.
From this textbox the user can select which measurements to log during an analysis. The selected measurements will automatically appear in the textbox to the right, within the same frame. If the user was to change his or her mind and remove one of the measurements this is done simply by pressing the measurements in the textbox and it will disappear from the textbox and be excluded from the simulation. When the user has selected all measurements which he or she wants to be logged the simulation is started by pressing the *Run Analysis* button. The application will then run the replay and log all the measurement values during the run. All of these operation are performed in the *2. Select and Analyze Measurements* frame.

Once the analysis has been completed each logged measurement is shown in the textbox, in the frame *3. View Measurements*. From here the user can select a measurement and press the button *View Measurement* to view the data. Basic information such as; min and max values for the magnitude-, X-, Y- and Z-length and in which timestep these occur will be shown. The user can also export the data if he or she wants to study the entire data set in greater detail. This is done by using the *Export Measurement(s)* button. The data will then be exported to an excel sheet, from where it can be plotted graphically.

The error handling of the *Post-Processor* tab is of similar nature as the *Mapping* tab and the *Pre-Processor* tab. That is, the application will point out when there is an issue, but it is up to the user to fix the issue.

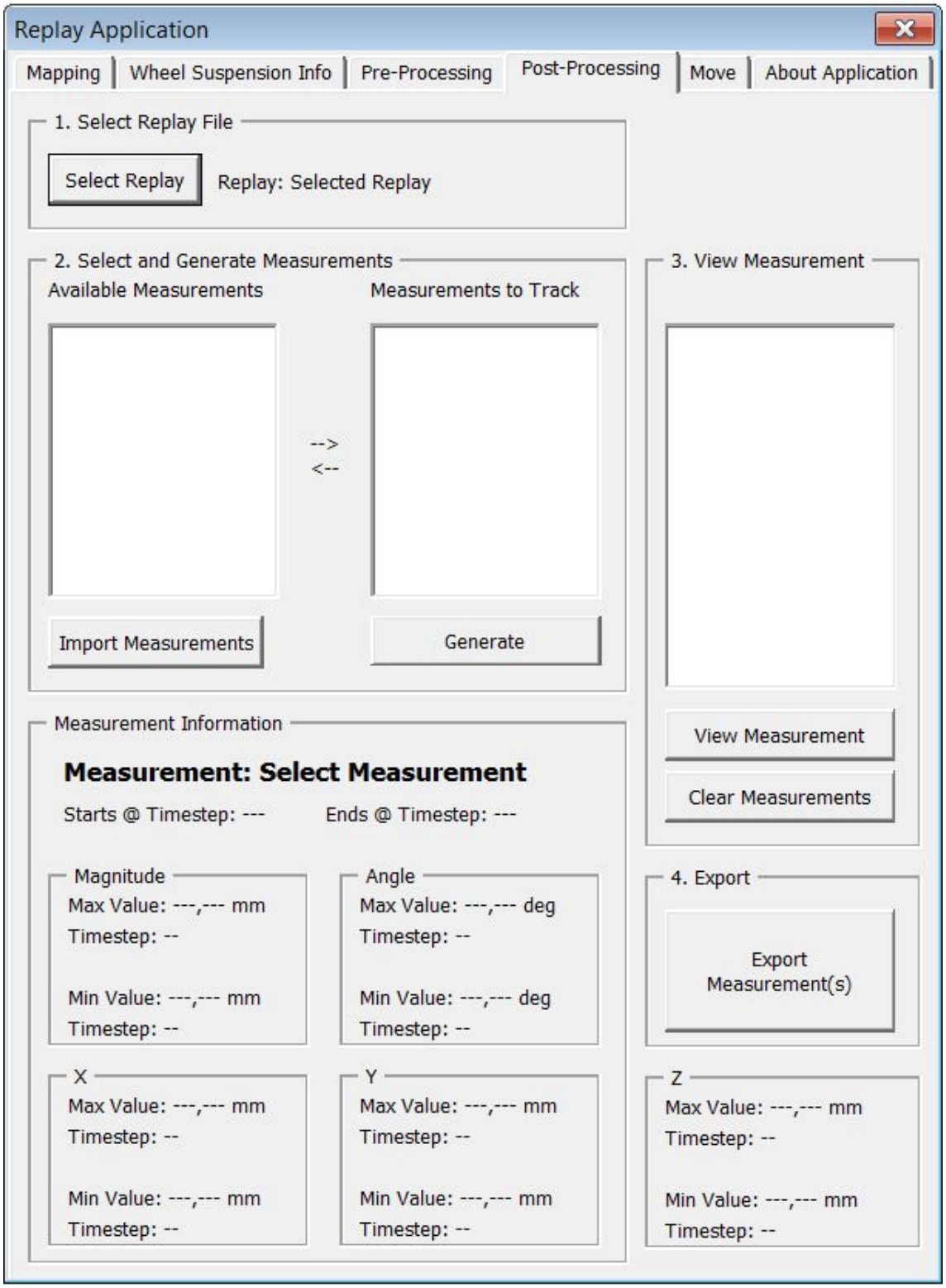


Figure 4.7: Illustrates the final version of the Post-processing tab in the GUI.

4.3.4 Features from detail design

Based on the feedback from the pilot study two major additional functions were added to the application. The first function is a table of information which shows various parameters that are of interest when developing a wheel suspension. By viewing this data the design engineer can quickly determine if the .res-file is of interest or not. The second function allows the user to move the wheel suspension model to a specific position from the simulation and keep this position when the replay file is closed. This allows the user to see how the vehicle behaves with different loads (where e.g. Curb + 2 is the vehicles position with two passengers and a certain amount of fuel in the tank). These positions are used by design to determine how the vehicle will look with different loads in place. Both of the functions were implemented in separate tabs which are called *Basic Wheel Suspension Info* and *Move*, respectively. Figure 4.8 illustrates the final version of the *Wheel Suspension Info* tab, while Figure 4.9 illustrates the final version of the *Move* tab.

As mentioned, the tab *Wheel Suspension Info* contains a summary of relevant parameters. In order to be able to use tab the user must first load a .res-file in the *Mapping* tab. Once this is done the tab can be used. The first step, textit1. Select Suspension, is to select front or rear suspension, the difference is the row *Rack Travel* which is used for the front suspension as it involves steering. In the second step *2. Set WLC + Z* the user sets a value which corresponds to a certain WLC + Z position (e.g. Curb + 2) and a tolerance limit. The tolerance limit is used to extract all timesteps which are within the limits from the .res-file. The values are then made selectable in the curtain menu marked *Timestep*. The parameters for the selected value are then plotted in the *Info* box which contains the following parameters:

- **WLC Z Coord**. The coordinate for the wheel center relative to the global CS, in millimeters.
- **WLC Z Travel**. How far the wheel center has moved from the design position in ADAMS/Car, in millimeters.
- **Toe Angle**. See Section 2.6.2, the unit is degrees.
- **Camber Angle**. See Section 2.6.2, the unit is degrees.
- **Damper Travel** How much the damper has compressed or extended from its design position, the unit is millimeters.
- **Damper Length** The length of the damper, it can be either extended or compressed, the unit is millimeters.
- **Rack Travel** The displacement or the steering rack, i.e. what angle the wheels have, the unit is millimeters.

The values can be viewed for four different wheel center positions:

- **Rebound**. When the wheel is in its lowest position.
- **Jounce**. When the wheel is in its highest position.
- **ADAMS/Car's DP**. When the wheel is in its design position in ADAMS/Car.
- **WLC + Z**. When the wheel is in the requested position.

The tab *Move* which is illustrated in Figure 4.9 allows the user to move the wheel suspension to a desired position based on a replay file. The selection of replay file is done through the button *Select Replay* in the frame *1. Select Replay*. The frame *Possible Time Steps* illustrates the timesteps which the replay file starts and ends with, hence a timestep in-between these values must be selected. The user then has two ways to move the wheel suspension. The first way is to move the wheel suspension to the position of a certain timestep, this is done in *2. Move To Timestep*. The second alternative is to move the wheel suspension to a WLC + Z position, this is done in the box *3. Move To WLC + Z* but requires more work. First of all a replay file must be loaded as the wheel suspension is to be moved to a certain position, rather than timestep. The .res-file can be loaded by using the button *Select .res-file*. Thereafter a position is entered in the box below, as are tolerance levels and rack travel (if desired). Once the user has decided which position to move the wheel suspension to he or she can simply press the *Move* button. If the user wants to move the wheel suspension to a new position it must first be repositioned in its design position, this is done by pressing the *Restore* button.

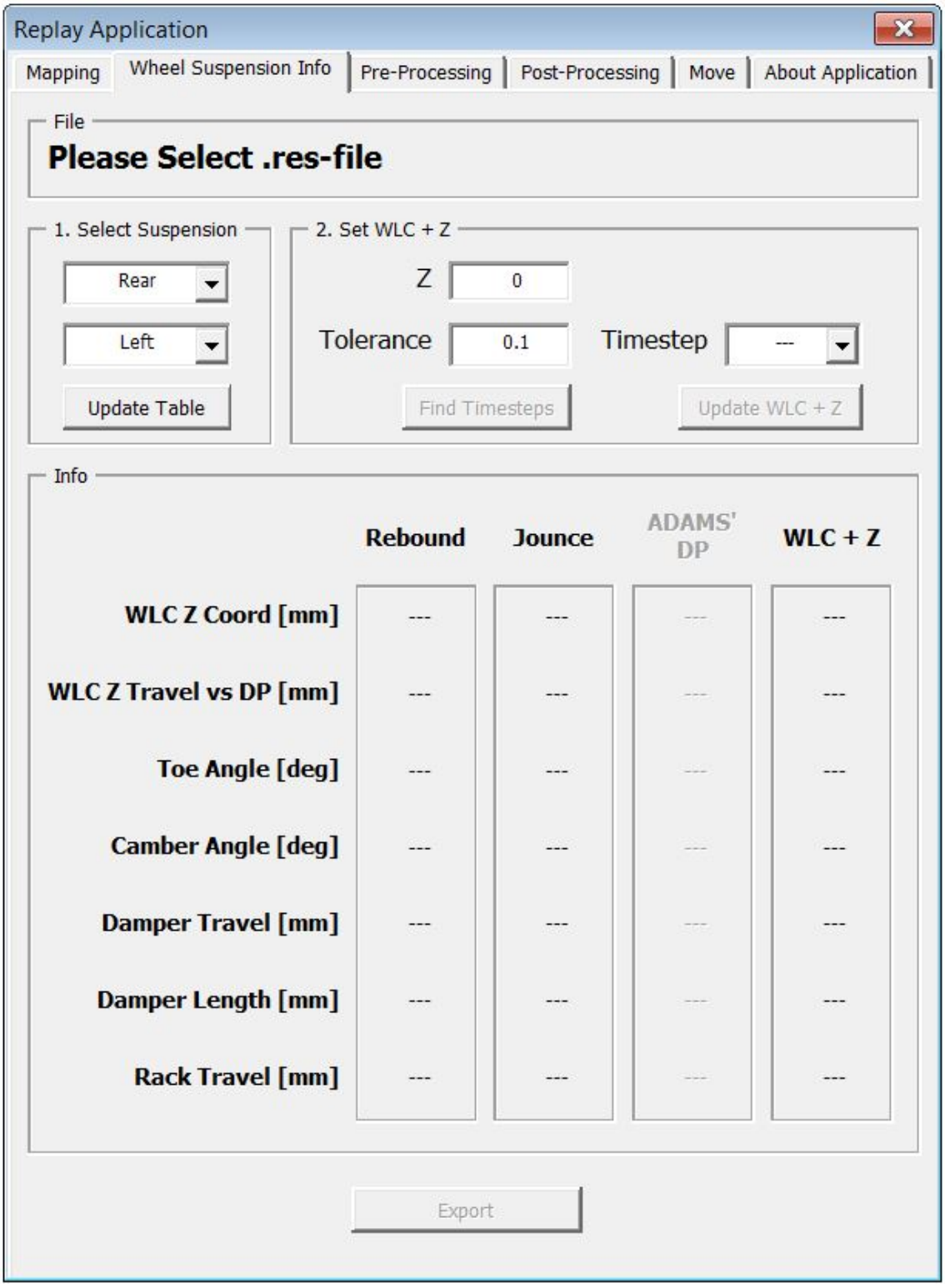


Figure 4.8: Illustrates the final version of the Wheel Suspension Info tab in the GUI.

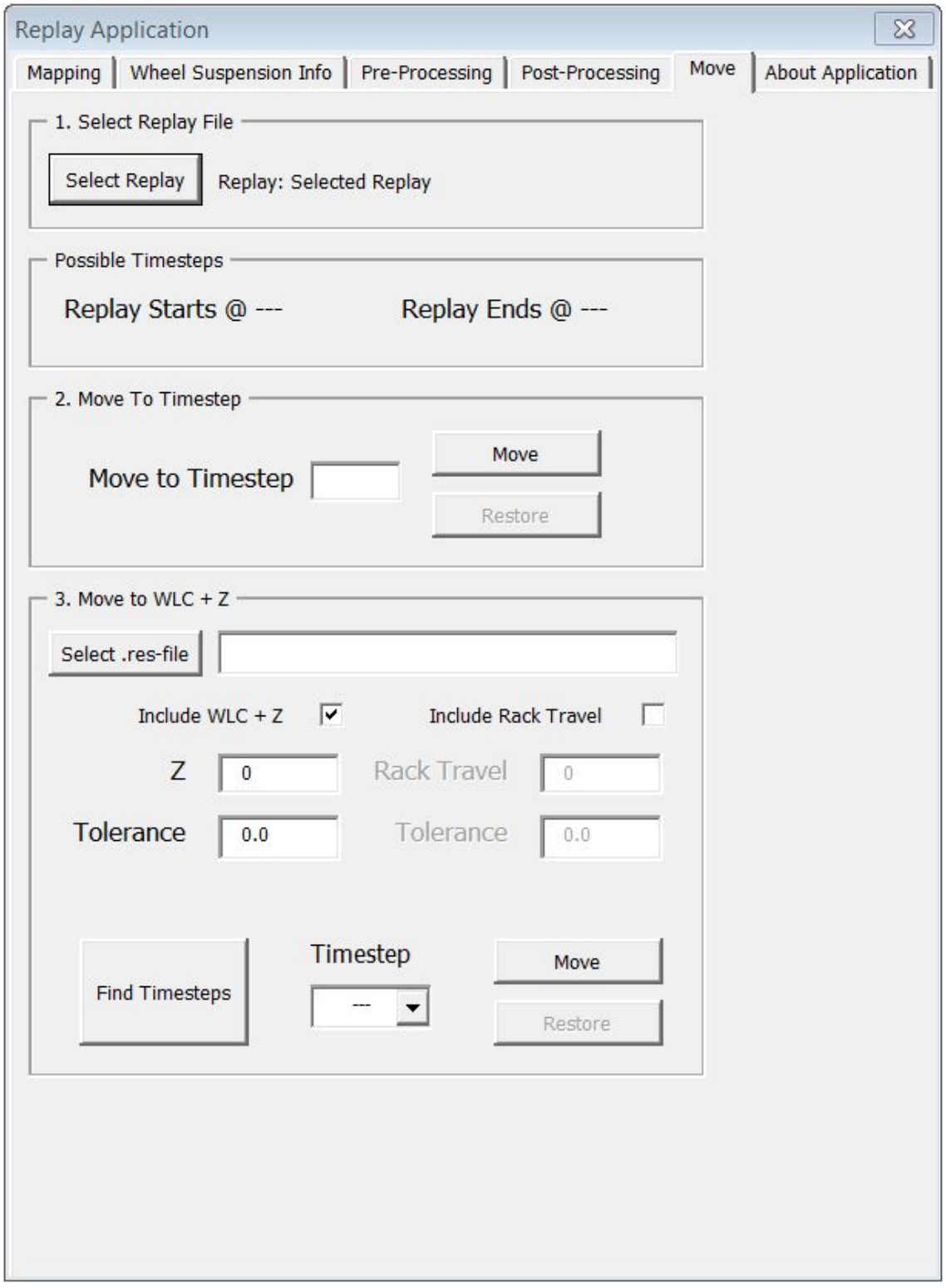


Figure 4.9: Illustrates the final version of the Move tab in the GUI.

4.4 Possible Additional Functionality

By implementing the new working procedure and thus also the application there are several opportunities which become apparent. The first, and possibly most prominent one, is the ability to generate design spaces by fixing one component and moving the others relative to this component. This becomes possible through matrix multiplication implemented in the code of the script. Where a component is fixed essentially by subtracting its translational motions from those of the other components and multiplying the transpose of its rotation matrix with its rotation matrix and then subtracting this matrix from those of the remaining components.

This project has been limited to focus on the rear wheel suspension of the SPA platform, as stated in Section 1.4. However, setting up the front suspension to work with the application is, in theory, only a matter of placing markers and requests in the necessary templates. Thus full scale implementation may be less of an hassle than initially believed.

Another opportunity, which requires work from Vehicle Dynamics, is developing combined drive-cases. As using the application to map motions enables use of the .res-file, instead of manually produced datasheets, Vehicle Dynamics can define a combined drive case which includes all the extreme drive cases required for a full clash analysis into one simulation. This becomes especially useful when implemented on the front wheel suspension where steering must be considered.

Another opportunity is the possibility of including flexible bodies in the analysis. This is however something which requires further work, both from Vehicle Dynamics and the Wheel Suspension Department. To be able to include flexible bodies the components in CATIA must be split into multiple sub-bodies. Further this means that vast numbers of markers and requests must be added to the templates, in order to do this in an efficient manner a clear naming structure, which accounts for the fact that each component will include several more measuring points than today, must be in place. One must note that planning and executing this work is no simple task completed over night. Instead it requires a complete re-design of the CAD-models used and the addition of several hundred markers and requests. In order to complete this task in an efficient way a smart and logical naming standard which accounts for requests that are not placed at a specific hardpoint must be developed. As a simplified example, consider the UCA, a relatively simple component in this case. It might be divided into three sub-bodies which together make up the complete component. Consequently a marker and a request which logs the motion corresponding to each sub-body of the UCA must be added in ADAMS/Car. This way the sub-bodies can be mapped to individual motions which correspond to the specific section of the UCA.

One final opportunity is to use the data transfer from ADAMS/Car to CATIA in other work performed by the Wheel Suspension department. One area where this application could be used during development of the hardpoint layout. Today this is an iterative process where a basic layout is set based on simulations and previous experience. Thereafter Vehicle Dynamics perform analysis on the layout and send the input back to the Wheel Suspension department, this process iterates until an acceptable layout is achieved. This process could

be automated in the same way as the data transfer process has been. By setting up statistical simulations, in ADAMS/Car, with the use of DoE hardpoint positioning could be analyzed w.r.t. the key parameters described in Section 2.6. Thereafter the resulting layouts can be automatically transferred to CATIA, by using the application, where packaging analysis is performed. The results can then be fed back into ADAMS/Car and new statistical experiments can be run if necessary. Setting up and controlling this loop would require the use of a software such as *modeFRONTIER* which couples third party software for seamless optimization loops [32].

5
Implementation Plan

This chapter contains a brief description of further work that must be done prior to implementation and to add additional functionality.

Once the master thesis project has ended the application will be handed over to the Wheel Suspension department. Near the end of the project meetings were held with representatives from the Wheel Suspension department, Vehicle Dynamics and IT Support regarding implementation of the application. In these meetings discussions were held on what must be done next and which department will be responsible for doing this. The tasks were split into short and long term implementation, where short term implementation consisted of tasks which must be carried out prior to implementation:

- A naming standard for markers and request must be established. The naming standard must include a prefix saying that the request is to be used for the application, it must include information on which instance it belongs to and where it is placed. This task will be handled by the Wheel Suspension department and Vehicle Dynamics.
- Once the naming standard has been set markers and requests must be placed in all affected templates. This work will be carried out by Vehicle Dynamics.
- The subroutine which is called by the requests must be compiled on a central server in order to work for all users at the company, this work will be handled by IT-support.
- Code maintenance will be handled by the Wheel Suspension department.

The points under long term implementation relate to additional functionality and application development, they ware as follows:

- Pre-study into how flexible bodies can be implemented. This includes investigating how CAD-models can be re-designed and how markers and requests are to be placed in ADAMS/car to achieve the desired results.
- Set up an automated design loop for hardpoint optimization by linking CATIA and ADAMS/car through software such as *modeFRONTIER* and by setting up statistical experiments in ADAMS/car, as explained in Section 4.4.

6
Discussion

This chapter contains the authors discussions regarding a range of subjects. First the working methodology used during the project is discussed, thereafter the proposed working methodology (developed in the project) is compared to the current working methodology for performing a packaging analysis of critical components. Further the application is discussed, as is the master thesis cluster, industry trends and future opportunities before the chapter is closed with a discussion regarding the goals of the project.

6.1 Development Methodology During Project

The methods used in this master thesis project differ some from the "common" product development master thesis projects, where most follow Ulrich & Eppinger's [29] proposed methods closely. In this project their methods have been used to some extent, the main working procedure which is described in Section 3.4 is largely influenced by them, however the *Concept Generation* and *Concept Selection* steps have been removed, or rather integrated in other parts of the process. The reasoning behind this is that since this is a software development project it simplified the development efforts. Software development projects work with significantly shorter design loops which end with quick n' dirty prototyping as soon as the code is compiled. Due to this concepts have not been developed in the same way as in "conventional product development" where a set of concepts are developed and screened against criteria. Instead one could claim that for software a new concept is developed every time a change is made in the code, meaning that in essence thousands of concepts have been developed. Of course documenting every change as a new concept and then screening them against criteria would be redundant. Instead each concept can be tested immediately by compiling the code, this means that as soon as a better solution is found it replaces the previous one, leading to a fundamentally different working procedure than during "conventional product development", with significantly shorter design loops.

Consequently a new working procedure requires a new methodological approach. For this the V-model has been used. This is a method which is found in most development efforts regardless of if it is on purpose of not. As described in Section 3.4 the principle is to break a system down into smaller parts at multiple levels. This approach was quite suited for the challenges faced during this project. Each part of the data transfer process had its own unique challenges which required its own approach, therefore it was beneficial to be able to break the application down into separate modules. Normally interface definition between modules can be a quite cumbersome task, this was not the case as the interface only consisted of input and output data, usually in the form of strings. So by breaking the

application down into small modules which could be developed independent of each other the work could be carried out in an efficient way.

6.2 Working Methodology Comparison

The two working methodologies, the current one and the proposed one, are quite similar as one aim of the project was to not change the process radically. The big difference lies in the time required to perform a design loop, as stated in Section 4.1 time saving of roughly 50 hours/critical component are achievable. The precision has also been increased due to a number of reasons. First, as all timesteps from a simulation can be used as measuring points. Second, by transferring motions from ADAMS/Car the generic rules used can be replaced by actual motions which take bushing stiffness into account. The precision will increase further as additional functionality such as flex bodies are added. Moreover, there is no additional knowledge required to use the proposed working process. Neither is there a need to compare the two processes in depth as this has already been done in Section 4.1 and the benefits of the proposed working process are clear. However, one could ask how much of an improvement the proposed working process is? There is no simple answer to that but the current benefits are stated above. If one is to be harsh it could be said that the application has more to offer regarding accuracy. Although most of the flex is accounted for by the bushings, which are included, there is still some flex in the components. Therefore including flex bodies would offer increased precision. On the other hand one must consider what is reasonable to deliver after 20 working weeks and also what is "good enough for now". By this we mean that in order to include flexible bodies far more work has to be done than what is reasonable for two master students in 20 weeks, see Section 4.4 and Section 5 for a more detailed description of what must be done. Regarding the expression "good enough for now" one must realize that starting from somewhere and implementing incremental development steps is more beneficial than trying to implement every functionality before releasing the software as this would delay the release significantly. Enjoying reduced leads times for longer is one benefit of this. Further, releasing the application earlier also give the users an opportunity to learn how to use it and realize what functionality must be changed or added in a way which is not possible in any other environment.

6.3 Application Design

The GUI of the application has not undergone any major changes regarding the layout during it's development. The general structure of the application and each tab has been the same, apart from the two tabs *Wheel Suspension Info* and *Move*, which were requested during the pilot study. We were able to set a design from the beginning which was close to the end result and implement a majority of the features early in the development, there are two main reasons for this.

The first reason is that a close dialogue was kept with one of the future main users of the application during the whole development process, through this dialogue daily discussions were held with the main user. This enabled us to identify and fulfill some basic user needs

in the early stages of the development process. If instead of this dialogue interaction would have taken place only a few times, the result would have been an application with less functionality and a GUI which would not suit the Wheel Suspension department's needs as well. In order to achieve the same amount of functionality as has been achieved, the workload towards the end of the project would have needed to be increased many-fold. Due to the regular contact, the workload could be spread out more evenly over the entire project. Moreover, the dialogue reduced the amount of iterations required for a feasible solution to be found. A scenario where this became apparent was when we proposed a solution to an issue and the main user could give quick feedback to weather the solution was suitable or not and come with additional suggestions, thus eliminating several design iterations. Due to the dialogue we could also gain deeper knowledge in how a wheel suspension works and thus better understand the needs of the Wheel Suspension department. Which in turn enabled development to begin closer to the end result.

The second reason is the large number of design loops ending with a quick n' dirty prototyping, these loops were used to test features and designs. This means that before realization of a feature or a design several alternatives had been tested in order to find the most suitable solution. From the design loops knowledge was gained which came in handy during discussions with the main user. The output from the knowledge gained during the design loops and the discussions were solutions which could meet user needs during the first implementation. As a result of this most functions in the final version have not needed refining, they were finished as soon as they were implemented.

The overall design of the application follows the structure of a book. Which has been appreciated by the main users. When using the application the user works from right to left and top to bottom, the tabs are also structured from right to left w.r.t. the workflow. This process follows the natural working process at the Wheel Suspension department even though all steps are not required to be carried out in sequence. However, since the application is more or less based on the current working process of the Wheel Suspension department, it requires that the user knows what he or she did in the previous steps. As stated each tab design follows the book design where the user starts in the upper right hand corner and ends at the bottom left hand corner of the tab. However, Volvo Cars have wheel suspension development in China as well and a book is read differently compared to Sweden. But since each step in each tab is numbered, it should be easy to understand the process even if the book logic does not work in China.

From both the main user and the pilot study simplicity and ease of use was emphasized as a key issue. Therefore providing sufficient error handling with explanations of the error has been important. The main reason to supply error handling is to ensure that the application does not crash when the user does something which is not intended to be done. If an error occurs the application will point out what has gone wrong, then it is up to the user to fix the issue. The reason to why the error messages point out what has gone wrong instead of just stating that something has gone wrong is that the issue are easier to address if the user is informed where he or she made a mistake. A mistake can come easily especially for inexperienced users, be it mapping of an instance to an incorrect motion or something else. It would be possible to take the error handling one step further and enable automatic

fixing of the issue at hand, however error handling of that caliber would require deeper programming knowledge and longer development time.

6.4 Application Properties

One feature which the application is dependent on is how an replay file is defined and created in CATIA. If the process of creating a replay file, or the structure of the replay file code, is changed the application may no longer function. However, since the replay is an established feature in CATIA and has existed for a long period of time the risk of fundamental changes taking place is quite small. If there is a structural change in the replay file the code would need to be updated accordingly, this requires great effort as the routine for creating a replay file is the largest and most complex routine in the application. The application is also dependent on the general structure of, and the naming convention in, the .res-file. If a change in the naming convention or structure of the .res-file takes place the probability that the application will malfunction is high. However, updating the application to follow a new naming structure is quite simple. In most cases this simply involves the updating of *IF-statements* to fit the new naming convention. A change in the general file structure is however a more complex matter, as the data extraction process is heavily dependent on the file structure described in Section 2.5.3 a change will render the application useless. Fixing this would require an entirely new routine. The probability of such a large structural change in the .res-file taking place is however small as it has remained the same for over 20 years. Therefore the application can be regarded as robust w.r.t. structural changes in CATIA and ADAMS/Car.

In order for the application to produce correct results the models in CATIA and ADAMS/-Car must be set up in a certain way. In ADAMS/Car the markers and requests must be set up according to specification, i.e. markers must log the motions of the thought body CS while the requests must call the conversion routine which converts Intrinsic 3-1-3 rotations to Extrinsic 1-2-3 rotations. If any other rotation sequence is inserted in CATIA the motions will not be accurate and thus make the packaging analysis obsolete. In CATIA the sensitivity relates to the placement of the body CS. When creating a replay file all body CS' must be placed in the absolute origin, if this in not done the replay file will not show the true movements of the instances. This is the area which the application is least robust in. Markers and requests are placed in templates therefore the work is only done once, CAD-models however are created more often and thus run a greater risk of being set up in an incorrect manner. It could thus be of interest to incorporate a function within the application which verifies that all body CS' are placed correctly. Further, a similar functionality could be implemented in ADAMS/Car for verification of marker and request definitions. Such functions will however need to be implemented in the future as they do not fit in the scope of this project.

As the replay file is based on motions transferred from an ADAMS/Car model rather than mimicking this motion in CATIA the resolution can be increased, in part because all timesteps can be handled as measuring points but also because new data, such as bushing stiffness can be included in the analysis. This becomes possible by automating the data transfer as handling such a large amount of information manually would not be a feasible

solution. In theory the precision is only limited by the number of timesteps which can be set in ADAMS/Car, however additional timesteps mean longer computation time and longer analysis time. Consequently it could be of interest to investigate where a good trade-off can be found between precision and analysis time.

Since data is being transferred from one software to another, with calculations being performed in the process information may be lost. However, the data can be transferred with a precision of thirteen decimal points, which is more than enough to perform a packaging analysis. Even though reducing the amount of decimal places during the data transfer may speed up the process slightly this was not done. The reason being that a small change in a rotation could lead to a mismatch between the movement of the two models (due to the lever effect resulting from the body CS placement described in Section 2.4.3 and thus reduce accuracy of the packaging analysis.

6.5 MSc. Thesis Cluster

As stated in Section 1.6 this master thesis in included in a cluster of three master theses. The thought behind this kind of cluster is to be able to handle larger and more complex projects. As students we can only speculate in the company benefits but we believe that this kind of collaboration, if done right, offers greater value for the company as larger projects can be undertaken. However, setting up master theses which are to collaborate closely may require a different approach when outlining the theses. I general the outline of a master thesis can be quite vague and thus be adapted to the competences of the students who are selected to carry out the work. Successful close collaboration usually requires well defined frameworks, something which seldom exists in a master thesis outline. Therefore it might be more important to identify a certain profile which is required to carry out the work, meaning that the students are adapted to the master thesis and not vice versa. On the other hand this might result in less master theses being carried out, if the required profile is not found.
As students this collaboration has given us an excellent opportunity to broaden our network, both with other students and within the company, Volvo Cars. The cluster has also offered us a platform where common challenges, often regarding software, can be discussed and knowledge can be shared. This is an area where much can be gained by increased coordination and collaboration. Gathering master thesis projects which use the same software and setting up workshops whose purpose is to discuss software challenges could save other master thesis students valuable time as they could realize mistakes before they are made.

6.6 Trends in Industry

Volvo Cars R&D program is undergoing a transformation under the name of 202020, described in Section 1.1, where the goal is to shorten time-to-market. The objectives of this master thesis project is well in line with the objectives of the transformation as part of its main benefit is development time reduction. The choice to develop scripts in-house is also a general theme throughout the company. This strategy seems to be a result of

the current software strategy where current software packages are being interconnected through in-house developed scripts. This may very well be a cost issue at multiple levels. On one hand in-house development is cost efficient as it makes use of the resources which are already in place. However, contracting experts may cost more but could save time as the work would probably get done faster. This trade-off, cost vs. time, must be considered. Will the goals be reached in time? If so, then in-house development may very well be the way to go.

On a higher level one must consider the companies long-term software strategy. There are software OEMs who have begun developing integrated CAD & CAE solutions. In the long-term this may very well be something worth implementing when the time comes to replace the current CAD and CAE environments. This is of course not a change which takes place over night, nor it it an inexpensive affair. When taking such a decision the amount of knowledge which the company has built up within the current environment must also be considered, both in terms of employees and their expert knowledge but also in terms of in-house developed scripts and solutions. Switching software environments to this extent is an issue which must be handled at the very top of the company and will most certainly result in efficiency losses for at the very least least one year. Moreover software OEMs depend on the financing and collaboration of future customers when developing new software. Therefore getting involved early may prove to be a benefit in the long term as the software would, to some extent, be tailored to fit the needs of Volvo Cars. However, one must also consider the size of the company in such a situation. Volvo Cars is a relatively small player compared to the big automotive OEMs who are involved in such software projects. Some of the largest automotive OEMs even have the resources to develop their own software tools, something which Volvo Cars cannot do yet. None the less getting involved in software development projects may still be possible if doing so in collaboration with other OEMs. This could be done by creating a cluster of future customers for software OEMs who co-finance the project, subject to the ability to agree on what functionality the software should include.

6.7 Further Improvements & Future Areas of Use

This application has enabled automation of cumbersome tasks is an efficient way to help achieve the 202020 transformation, Section 1.1. Throughout the project we have seen that similar applications exist in other areas of the organization, either developed in other environments or for other purposes. Therefore it is reasonable to assume that there are other departments within Volvo Cars who could benefit from using this application. One of these could be the steering department, who also work with replay files in CATIA. Another may be the engine department who monitor engine movement during driving. The later already use similar applications in other environments but could none the less be interested in this application. Further it may be of interest for the Wheel Suspension department to take part in the script developed in other departments. When reasoning this way it becomes apparent that some kind of forum where departments can demonstrate and discuss their internally developed scripts would be beneficial, this way the cross-functional communication will increase and as the level of automation grows unnecessary work is

bound to be re-done at separate departments if no dialogue is kept.

When looking closer at the application the ability to transfer motions from ADAMS/Car opens up the possibility of including flexible bodies in the simulation. However, as mentioned in Section 4.4 this requires additional work. But before this work is carried out a study regarding which components to model as flex bodies in CATIA must be carried out. For a majority of the components in the wheel suspension body flex is of a secondary nature as they are designed with stiffness requirements in mind, however components such as the anti-roll bar and leaf spring have considerable body flex as elastic deformation is part of their functionality. By establishing a set of components to be re-modeled to include body flex much time and effort can be saved compared to re-modeling all components of the wheel suspension.

As mentioned in Section 4.4 the application can also be used when developing the hardpoint layout for a new wheel suspension. This is also an opportunity which requires a great deal of work to function in an efficient way. Setting up statistical experiments and logging the results is a good way to log and visualize the expert knowledge used for setting the initial hardpoint layout. As the data is visualized in such a way that the placement of hardpoints correlates directly to a key parameter, e.g. camber angle, the impact of moving a hardpoint can be visualized in a quick and efficient way. This in turn means that expert knowledge can be spread to the rest of the department without any extra work. One challenge with this working procedure is the output from CATIA. It must be decided what the output is to be and how this output will be utilized and fed back into ADAMS/Car. When a certain setup leads to a clash an automated way of identifying which hardpoints affect the clash must be developed. Further an algorithm which adjusts the span of which these hardpoints can move during the statistical experiments must also be developed. When this design loop can be set up in an efficient way the time saving potential is huge during the early phases of development.

6.8 Project Goals

The application is able to transfer data from ADAMS/Car to CATIA by using the .res-file, which is a standardized file format. Replay files can be created and certain post-simulation analysis functions have been implemented, however more is to come. There is no information loss during the data transfer as the data is transferred with more decimal points than what the packaging analysis is carried out with. Further the precision of the packaging analysis has been improved, as has the data handling capacity. The application is intuitive to use, based on user feedback, and follows the same working procedure as the current process, but allows for significant time savings. Thus it seems that all goals have been fulfilled. This is good but it raises some questions, could more have been added? Were the goals too unspecific?

The first question could have two answers, yes and no. Yes, as working more efficiently could, in theory, allow more functionality to be added. On the other hand development efforts take time, especially when it is done for the first time. In a scenario where all knowledge would have been gathered prior to project start the development time would be

reduced significantly. This is however not possible the first time a certain kind of work is carried out as the knowledge must be gained. No, as stated above, gaining the necessary knowledge take sits time, it is always easy to look back once you know something and say; "well that is easy". Therefore the extent of the project seems to have been set quite well.

The second question, regarding unspecific goal formulations, is true for some goals and less true for others. The goal to include post-simulation analysis is the most vaguely formulated one. As meeting the goal could mean implementing the smallest of post-simulation analysis or implementing 100 post-simulation analyses. Thus this goal could be reformulated. The goal to minimize information loss during the data transfer may seem vague but is complemented with a specific value in the requirement specification. Further, the goal enable replay file creation is equally specific as this goal can either be met or not, there is no scale in-between.

7
Conclusions

This chapter contains the final conclusions from the master thesis project.

First of all we can conclude that proof of concept, regarding data transfer, has been successfully proven. This in the form of an application which is ready to move into the implementation phase. We can also conclude that there is still work to be done and functionality to be added, which will ease the work of the wheel suspension department. Further the application cuts development time by roughly 50 effective working hours for packaging of a critical component, while increasing analysis precision, both by including more measuring points and by including additional data. Moreover the application has received good feedback regarding ease of use as it has an intuitive GUI and it fits the current working procedure.

The structure of the application is generic, meaning that it can be used to map motions and instances in general, not only wheel suspension instances. Consequently the application can be used by other departments who transfer motions from ADAMS/Car to CATIA, provided that markers and requests have been placed in their ADAMS/Car templates, as described in Section 3.4.2.1. Further the application can be regarded as robust w.r.t. format changes both in ADAMS/Car, the .res-file and in CATIA's replay file structure, as none of these file structures are likely to undergo radical changes. Due to the factor of human error placement of markers and requests in ADAMS/Car, along with body CS placement in CATIA are the two areas which the application is least robust in. The latter of the two being the most critical as this task is repeated more frequently.

Regarding hardpoint optimization this process has not been fully automated. The application will be one building block in such a process, there are however several other blocks which must be developed and integrated before this process can be fully automated.

The master thesis cluster has worked as a great opportunity to broaden personal networks, discuss challenges and share learnings. This is a concept which is worth developing as it could offer companies greater value from master thesis projects as larger projects can be undertaken.

8

Future Work Proposals

This chapter contains future work proposals based on the results of this master thesis project.

8.1 Flexible Body Implementation

As the application is designed to enable handling of flexible bodies further studies into this area are recommended. The authors believe that a pre-study regarding this could be carried out as a master thesis. The thesis would begin with a literature study into the area and could also include a benefit analysis, i.e. how much accuracy can be gained and which components are worth re-modelling, it must also include re-modelling of one, or a set of CAD instances within the wheel suspension. A study could be done where different ways of splitting components into sub-bodies are compared w.r.t. increased precision, modelling complexity and time consumption. The thesis would also include placement of markers and requests in ADAMS/Car which correspond to the different modelling approaches in CATIA. Results could be visualized through a plot with performance index and "mesh size" on the two axes.

8.2 Hardpoint Optimization Through CAD & CAE Loop

Enabling hardpoint optimization would require a closed automated design loop between ADAMS/Car and CATIA, where statistical experiments could be set up in ADAMS/Car, the resulting wheel suspension layouts would be analyzed in CATIA w.r.t. packaging. Further the results from the analysis will be fed back into ADAMS/Car where to positioning span of concerned hardpoints is changed, this is described in further detail in Section 4.4. The thesis could include setting up statistical experiments in ADAMS/Car and creating an automated design loop using software such as *modeFRONTIER*.

8.3 Application Improvement

The application could be refined further in order to improve the user experience and add additional functionality. In both the Pre-processor and the Post-processor it would be good if the user could visualize diagrams over the data without exporting it to excel. Another point to improve would to be to allow the user to perform a wider range of analysis instead of just being able to measure the distance between instances. For instance, it would be preferable to see where the clash occurs instead of just seeing that there is a clash. The

GUI also has potential for further improvements, both regarding the information shown and the functions included, in order to ease the design engineers work. Further it could be of interest to improve the code itself in order to maximize the efficiency of the application. This could be done either through a master thesis, bachelor thesis or a summer job for students at VESC, depending on what goals are set. It would be suitable if the Wheel Suspension department or IT, were responsible for the work. As the application has not been a natural part of the development process it may not be suitable to carry out this work in the near future.

8.4 Combined Drive-cases

By automating the data transfer process the possibility of creating combined drive-cases has emerged. This would be done by Vehicle Dynamics, possibly as a master thesis. Combined drive-cases could be used for more advanced clash analysis or creation of wheel envelopes. It could also be used to minimize the number of runs Vehicle Dynamics perform. All drive-cases required for packaging analysis, by the Wheel Suspension department, could be included in one simulation where they are queued after each other. Design engineers could then create replay files for a certain set of timesteps to look at a certain drive case.

Bibliography

[1] Volvo. Pd transformation 202020. Internal Material.

[2] Mikael Sellergren. Mechanical design engineer. unpublished.

[3] Kanishk Bhandani and Joakim Skön. Balancing of wheel suspension, performance and weight. Master thesis, Chalmers University of Technology, 2016.

[4] Robin Larsson. Structural topology and shape optimization. Master thesis, Chalmers University of Technology, 2016.

[5] Harald Hasselblad. Optimization culture arena, April 2016.

[6] Volvo Group. Volvo group history, April 2016.

[7] Christer Olsson and Henrik Moberger. *Volvo, Göteborg, Sverige*. Norden Publishing House, St. Gallen, 2. uppl. edition, 1997.

[8] Ford Motor Company. Ford media center, April 2016.

[9] Volvo Cars. Volvo cars financial results, April 2016.

[10] Steven C. Wheelwright and Kim B. Clark. *Revolutionizing Product Development: Quantum Leaps in Speed, Efficiency and Quality*. Free Press, New York, 1992.

[11] Kuang-Hua Chang. *Product Performance Evaluation with CAD/CAE*. Academic Press, Boston, 2013.

[12] David G. Ullman. *The Mechanical design process*. McGraw-Hill, Boston, 4. edition, 2010.

[13] David M. Anderson. *Design for Manufacturability*. CRC Press, 2014.

[14] C. T. Leondes and ScienceDirect Archives (e-book collection). *Concurrent engineering techniques and applications*, volume 62.;62;. Academic Press, Inc, San Diego, California;London, England;, unit kingdom edition, 1994;2014;.

[15] Avelino J. Gonzalez and Douglas D. Dankel. *The engineering of knowledge-based systems*. Prentice Hall, Englewood Cliffs, N.J, 1993.

[16] Hong-Seok Park and Xuan-Phuong Dang. Structural optimization based on cad & cae integration and metamodeling techniques. *Computer-Aided Design*, 42(10):889–902, 2010.

[17] Robert A. Adams and Christopher Essex. *Calculus: a complete course*. Pearson, Toronto, 8. edition, 2013.

[18] Anders Boström. Rigid body dynamics. Technical report, Chalmers University of Technology, October 2015. Course material for MAA092.

[19] James Diebel. Representing attitude: Euler angles, unit quaternions, and rotation vectors. Technical report, Stanford University, 2006.

[20] Daniel Molin. *CATIA V5 Basic Kinematics*. Volvo Cars Corparation, 2015.

[21] Dassault Systems. *CATIA Documentation*, 2014.

[22] Michael Blundell and Damian Harty. *The Multibody System Approach to Vehicle Dynamics*. Elsevier Butterworth-Heinemann, 2004.

[23] Daniel Molin. Supervisor. unpublished.
[24] MSC Software. Adams/car simulation environment for vehicle design and testing, April 2016.
[25] Technical Committee ISO/TC 22. *ISO 8855 - Road Vehicles - Vehicle Dynamics and Road Holding Ability*. International Standardization Organization, 2011.
[26] Bernd Heissing and Metin Ersoy. *Chassis handbook: fundamentals, driving dynamics, components, mechatronics, perspectives*. Vieweg + Teubner, Wiesbaden, 1st edition, 2011.
[27] Pamela J. Waterman. Closing the cad to cae gap, March 2014.
[28] G. P. Gujarathi and Y. . Ma. Parametric cad/cae integration using a common data model. *Journal of Manufacturing Systems*, 30(3):118–132, 2011.
[29] Karl T. Ulrich and Steven D. Eppinger. *Product design and development*. McGraw-Hill/Irwin, New York, NY, 5th edition, 2011.
[30] Richard Stevens. *Systems engineering: coping with complexity*. Prentice-Hall Europe, London, 1998.
[31] Fredrik Sjögren. Msc adams/car specialist. unpublished.
[32] ESTECO. Mode frontier, June 2016.
[33] Dai G. Lee and Nam P Suh. *Axiomatic design and fabrication of composite structures: applications in robots, machine tools and automobiles*. Oxford University Press, New York, 2006;2005;2012;.

A

Wheel Suspension Types

The general trend within wheel suspension development has been to uncouple the design in order to increase tuning possibilities [26], in product development literature this is referred to as axiomatic design [33].

A.1 Wheel Suspension Trends

There is no shortage of wheel suspension configurations used throughout the history of car manufacturing. Wheel suspension systems can be classified in different ways, a reasonable place to start is by dividing the system into front and rear wheel suspension systems, in theory most systems could be used for either axle. However, this is not the case in reality due to various customer demands and requirements, e.g. packing space, steering, ride comfort and the list goes on and on. The second classification which the authors set concerns pricing, instead of a Boolean operator this classification works as a continuum. By adding a third classification which will be the ability to tweak kinematics, handling and ride comfort attributes an explanatory plot such as the one in figure A.1 can be created. Figure A.1 features the most common rear wheel suspension types which are currently in use. Rigid axles are most commonly found on trucks and off-road vehicles, while torsion crank, longitudinal link and semi-rigid axle systems are common on small and lower mid-size cars. The cluster positioned in the top right-hand corner of the plot are most commonly found on upper mid-size and premium cars.
The choice of wheel suspension type depends on the targeted market segment, customer base and vehicle type. As mentioned, OEMs who manufacture small and mid-size cars for the lower price segments use either semi-rigid axle suspension, torsional crank axle suspension or longitudinal link suspension systems. The main reason being the sensitivity to price in this segment. OEMs in the premium segment, where customers are less sensitive to price, more commonly work with double wishbone suspension, multi-link suspension or five-link suspension systems. The main reason being the required ride comfort, in the premium segment, which is easier to achieve with these three suspension systems. Also, customers in the premium segment are less sensitive to price and thus prone to pay more for increased ride and/or handling attributes. OEMs must also take packaging space, cargo space, compatibility with different powertrains - as suspension system must work with e.g. both FWD or RWD and AWD, depending on the OEMs offerings - and what types of vehicles the suspension must fit - some OEMs need to use the same suspension system on wagons, sedans, coupes and maybe even SUVs - into consideration. In other words selecting and designing a wheel suspension system consists of making a set of trade-offs so that the chosen system fits the requirements of the segment where it is to be sold.

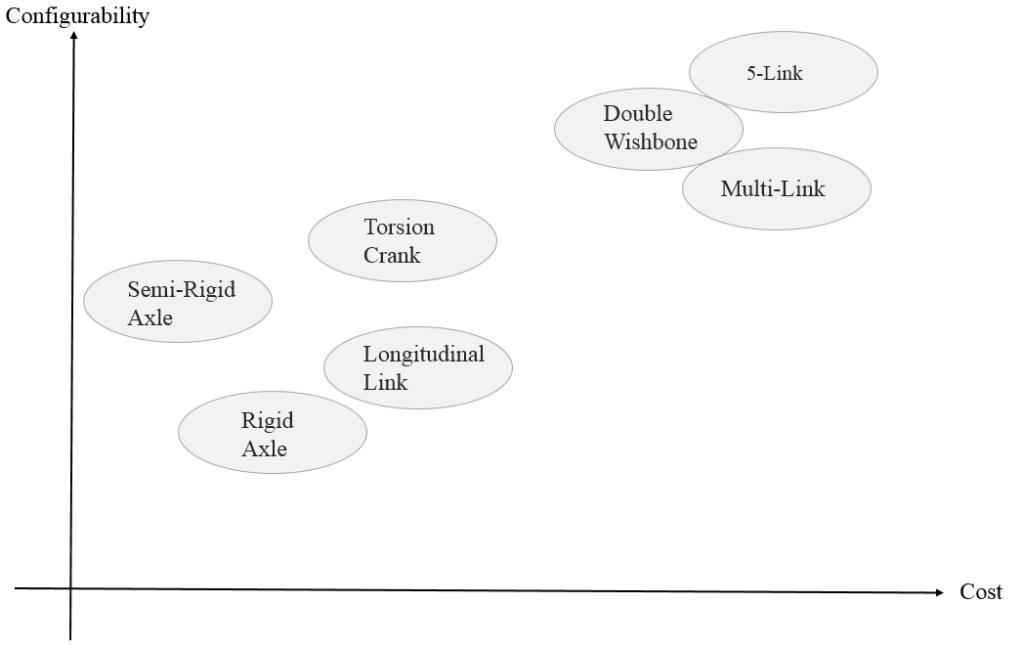


Figure A.1: The graph features the six most common rear wheel suspension systems according to Heissing and Ersoy [26]. The authors of this report have added the five-link suspension system as well, as it is believed to grow in significance.

A.2 Rigid Axle Suspension Systems

Connecting a wheel pair with a rigid axle is the oldest solution, it originates from the horse carriages which preceded cars. A rigid axle system offers a low cost solution with a robust design and high load capacity. This comes at the expense of ride comfort due to a large unsprung mass and the fact that the motion of the two wheel in a wheel pair is directly coupled. Today rigid axles are most commonly found in off-road vehicles and larger trucks, where ride comfort is of secondary interest. Figure A.2 illustrates a rigid axle wheel suspension system.[26]

A.3 Semi-Rigid Axle Suspension Systems

The obvious step in decoupling the design was to reduce the effect which the motion of one wheel has on the other. Semi-rigid axle suspension does just this, the wheels are still connected by a cross member, however elastic deformation of the cross member reduces the effect one wheel has on the other. The deformable cross member can be attached directly at the wheel centers, or trailing arms, at different positions, see figure A.3. By attaching the cross member along two trailing arms it will be subjected to torsional loads, with varying magnitude depending on the distance to the wheel center. This kind of setup the authors of this paper believe to be what Heissing and Ersoy [26] refer to as "torsional crank suspension". The main benefits of semi rigid suspension systems include robustness with large loading capacity, flat and compact packaging and good anti-pitch properties while being a low cost solution. The drawbacks include limited ride comfort,

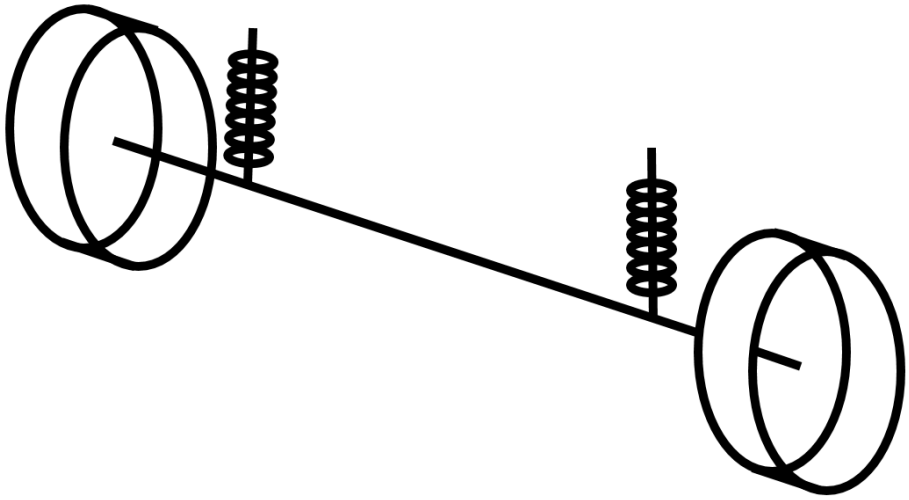


Figure A.2: Illustrates the rigid axle wheel suspension.

stress concentrations in welds connecting stiff and deformable parts and poor lateral force resistance. [26]

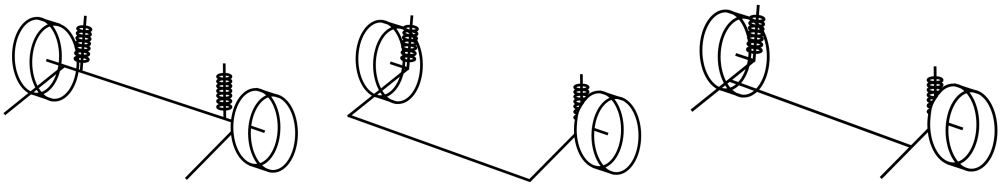


Figure A.3: Illustrates the three types of Semi-rigid torsional crank suspension systems.

A.4 Independent Suspension Systems

After the introduction of semi-rigid axles the next step in development was to continue decoupling the wheels, meaning that the movements of one wheel are not transferred to the other. This means that each wheel is assembled to the car body, or sub-frame, individually. The general advantages compared to rigid and semi-rigid solutions include reduced unsprung mass, independent wheel movements, possibilities to optimize the wheel suspension both w.r.t. kinematics and elastokinematics, also the handling of improved roadway vibrations and acoustics becomes easier. Independent suspension systems are however more expensive and significantly more complex than rigid and semi-rigid suspension systems. Neither are they as robust and since the wheels are decoupled an anti-roll bar connecting the wheels is required to equalize the loading during cornering.[26]

Longitudinal Link Suspension
What the authors of chassis handbook [26] refer to as longitudinal link suspension systems

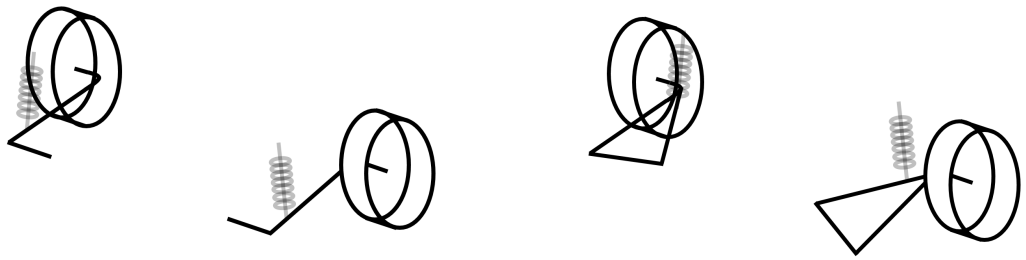


Figure A.4: Illustrates the trailing Arm Independent Suspension System (left) and the Semi-trailing Arm Independent Suspension System (right).

seem to be a collection of trailing and semi-trailing link suspension systems. These two suspension systems have quite different advantages and disadvantages, which makes the grouping quite questionable. The trailing link suspension system, figure A.4, is a fairly simple construction which requires little packaging space while allowing individual wheel control. The disadvantages are bad roll stability (so bad that an anti-roll bar is required), poor straightline driving properties and withstanding the impact of lateral forces, which easily cause oversteer. [26]

The semi-trailing link setup, figure A.4, works as a compromise between a trailing link and swing axle system, while enabling the optimization of kinematics by adjusting the incline and sweep angles of the trailing arm. The disadvantages of this setup include large camber changes during jounce and rebound, camber and toe angles are coupled and lateral forces result in toe out, which impact handling e.g. during cornering. [26]

Double Wishbone Suspension System
This suspension system uses three independent links, two of them are three-point-links shaped as a wishbone (thus the name) and one two-point link which is only used for steering. One of the three-point links is positioned below the wheel center and the other above, figure A.5. The exact positioning can be varied depending on the desired properties. The main benefits of the double wishbone suspension system include great design freedom as the roll-center and pitch axis can be chosen freely, while camber and track width changes, during wheel travel, are limited. The double Wishbone setup also offers high lateral stiffness as well as good ride and handling. The disadvantages include the requirement of large packaging space, complex and high cost manufacturing and the required use of a sub-frame due to the large forces applied to the chassis side. [26]

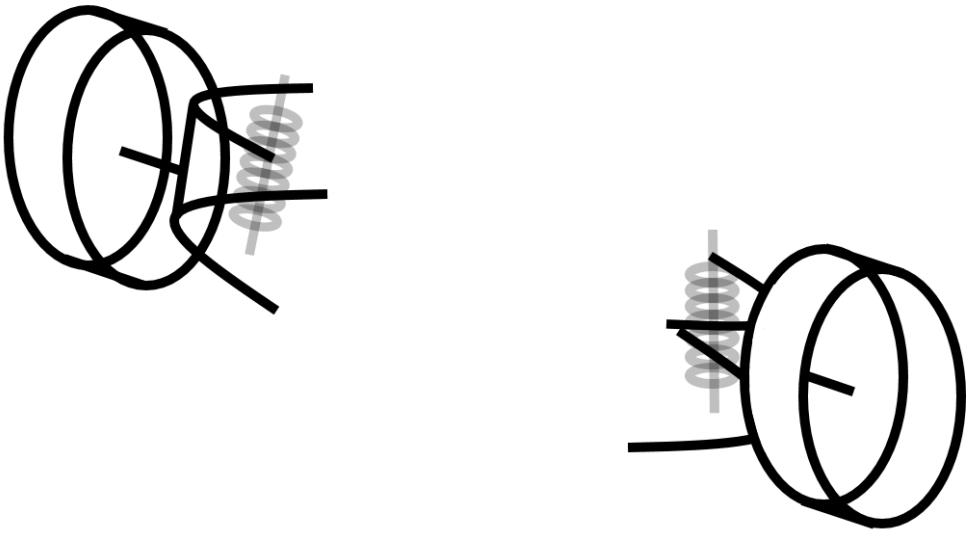


Figure A.5: Illustrates the Double Wishbone Suspension System.

Multi-Link Suspension System
A multi-link suspension system consists of three or more links, on the rear wheels one of the links usually is longitudinal. There are numerous multi-link configurations, as the links can be placed quite independent of each other. Most common is the use of three or four two-point links, however some OEMs use five links as well, figure A.6. According to theory [33] increasing the number of links further decouples the design, and thus increase the design freedom, up to a certain number of links where the system becomes over determined. The large design freedom allows OEMs to tune each parameter separately. This design freedom also means that OEMs can fine tune both ride and handling at a high level. However, as with the other independent suspension systems large packaging space is required, the cost for manufacturing is high, as is the complexity of the system.

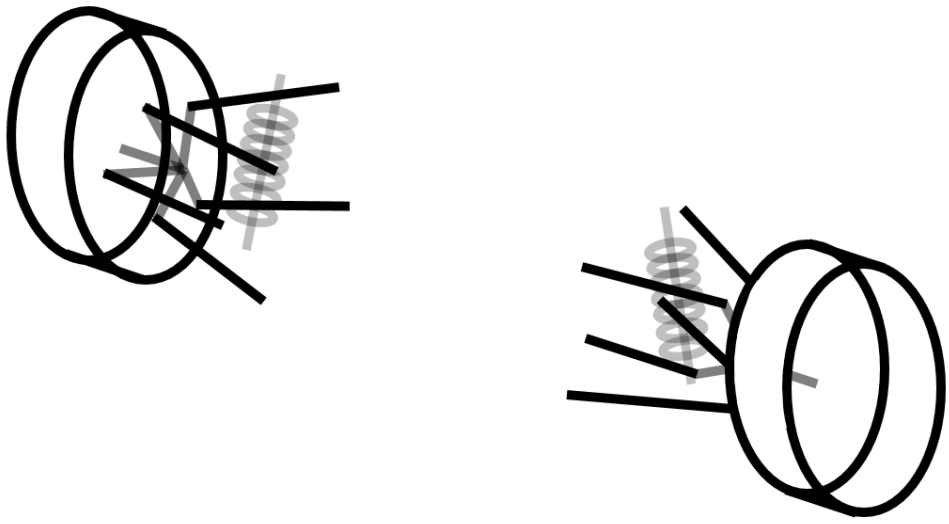


Figure A.6: Illustrates a Multi-link Suspension System with five links.

B

An example of "Create Replay" script

The following VB script creates a replay containing two parts, Object_A and Object_B. Sample 1, in part B, is connected to Object_A and Sample 2 is connected to Object_B.

```
Language="VBSCRIPT"

Sub CATMain()

'Part A,
'Declaration of replay
'Which parts/products that shall be included.

Dim productDocument1 As Document
Set productDocument1 = CATIA.ActiveDocument

Dim replayWorkbench1 As Workbench
Set replayWorkbench1 = productDocument1
                       .GetWorkbench("ReplayWorkbench")

Dim replays1 As Replays
Set replays1 = replayWorkbench1.Replays

Dim replay1 As Replay
Set replay1 = replays1.Add("Replay")

Dim product1 As Product
Set product1 = productDocument1.Product

Dim products1 As Products
Set products1 = product1.Products

Dim product2 As Product
Set product2 = products1.Item("Object_A")

Dim long2 As Long
long2 = replay1.AddProductMotion(product2)

Dim product3 As Product
Set product3 = products2.Item("Object_B")

Dim long3 As Long
long3 = replay1.AddProductMotion(product3)

'Part B, Definition of movement for each
'part/product in each timestep.

Dim arrayOfVariantOfDouble1(11)
arrayOfVariantOfDouble1(0) = 1.000000
arrayOfVariantOfDouble1(1) = -0.000000
arrayOfVariantOfDouble1(2) = 0.000000
arrayOfVariantOfDouble1(3) = 0.000000
arrayOfVariantOfDouble1(4) = 1.000000
arrayOfVariantOfDouble1(5) = -0.000000
arrayOfVariantOfDouble1(6) = -0.000000
arrayOfVariantOfDouble1(7) = 0.000000
arrayOfVariantOfDouble1(8) = 1.000000
```

```
arrayOfVariantOfDouble1(9) = 0.000000
arrayOfVariantOfDouble1(10) = 0.000000
arrayOfVariantOfDouble1(11) = 0.000000
replay1.AddSample 1, 0.0, arrayOfVariantOfDouble1

Dim arrayOfVariantOfDouble2(11)
arrayOfVariantOfDouble2(0) = 1.000000
arrayOfVariantOfDouble2(1) = -0.000000
arrayOfVariantOfDouble2(2) = 0.000000
arrayOfVariantOfDouble2(3) = 0.000000
arrayOfVariantOfDouble2(4) = 1.000000
arrayOfVariantOfDouble2(5) = -0.000000
arrayOfVariantOfDouble2(6) = -0.000000
arrayOfVariantOfDouble2(7) = 0.000000
arrayOfVariantOfDouble2(8) = 1.000000
arrayOfVariantOfDouble2(9) = -0.000000
arrayOfVariantOfDouble2(10) = 0.000000
arrayOfVariantOfDouble2(11) = 0.000000
replay1.AddSample 1, 1.0, arrayOfVariantOfDouble2

Dim arrayOfVariantOfDouble3(11)
arrayOfVariantOfDouble3(0) = 0.999919
arrayOfVariantOfDouble3(1) = 0.001549
arrayOfVariantOfDouble3(2) = 0.012601
arrayOfVariantOfDouble3(3) = -0.001008
arrayOfVariantOfDouble3(4) = 0.999083
arrayOfVariantOfDouble3(5) = -0.042799
arrayOfVariantOfDouble3(6) = -0.012656
arrayOfVariantOfDouble3(7) = 0.042783
arrayOfVariantOfDouble3(8) = 0.999004
arrayOfVariantOfDouble3(9) = 5.159741
arrayOfVariantOfDouble3(10) = -14.804487
arrayOfVariantOfDouble3(11) = 26.463163
replay1.AddSample 2, 0.0, arrayOfVariantOfDouble3

Dim arrayOfVariantOfDouble4(11)
arrayOfVariantOfDouble4(0) = 1.000000
arrayOfVariantOfDouble4(1) = 0.000217
arrayOfVariantOfDouble4(2) = -0.000678
arrayOfVariantOfDouble4(3) = -0.000223
arrayOfVariantOfDouble4(4) = 0.999957
arrayOfVariantOfDouble4(5) = -0.009257
arrayOfVariantOfDouble4(6) = 0.000676
arrayOfVariantOfDouble4(7) = 0.009257
arrayOfVariantOfDouble4(8) = 0.999957
arrayOfVariantOfDouble4(9) = 1.515281
arrayOfVariantOfDouble4(10) = -0.399632
arrayOfVariantOfDouble4(11) = 18.429855
replay1.AddSample 2, 1.0, arrayOfVariantOfDouble4

End Sub
```

C

Requirement List

This appendix contains the requirement list which begins on the next page due to formatting reasons.

Requirement Area	Requirement	Demand/ Wish	Description	Target Value	Verification Method
Data Transfer	Enable data transfer from ADAMS to CATIA	D	-	-	Verification runs
	Handle data transfer with available software within Volvo Cars	D	License cost must be taken into consideration as well	-	Verification runs
	Initiate data transfer in the CATIA V5 environment	W	-	-	Verification runs
	Handle data transfer in the CATIA V5 environment	W	-	-	Verification runs
	Develop CATIA V5 application to handle data transfer	W	-	-	Verification runs
Data Extraction	Extract motion data from ADAMS/car	D	-	TRUE	Verification runs
	Motions include bushing stiffness	D	-	TRUE	Verification runs

Data Extraction	Use standardized file format for data extraction	W	Data is extracted from a file which is already produced, i.e. does not add more work for Vehicle Dynamics	TRUE	Verification runs
	Extract motion data from ADAMS .res-file	W	the .res-file logs the results from each simulation and is always created automatically	TRUE	Verification runs
Data Insertion	The application converts rotations form B313 to S123	D	-	TRUE	Verification runs & comparison of motions between CATIA V5 & ADAMS
	Motion precision comply to accuracy used during packaging analysis	D	-	< 0.01	Measurement

Data Insertion	Motion precision exceeds the accuracy used during packaging analysis, as a safety margin	W	-	< 0.00001	Measurement
GUI	Reads multi-layer product tree	D	It is possible to read a product tree with instances at multiple levels.	TRUE	Verification runs
	Instances can be mapped to motions from ADAMS	D	CATIA instances can be assigned to motions from ADAMS. Regardless of which level in the product tree they are located.	TRUE	Verification runs
	A replay file is automatically generated	D	When the "Create Replay" button is pressed a replay file containing the mapped CAT instances and motions is generated.	TRUE	Verification Runs

GUI	The generated replay file is nameable	W	-	TRUE	Verification runs
	Possibility to save mappings	W	Mapped CAT instances and motions can be saved for later use by pressing the "Save Mapping" button.	TRUE	Verification runs
	Possibility to load saved mappings	W	Saved mappings can be loaded for use on the corresponding model, by pressing the "Load Mappings" button.	TRUE	Verification runs
	The application includes critical error handling	D	To ensure that the application does not crash if the wrong input is selected error handling is necessary.	TRUE	Verification runs

GUI	A CATIA instance can be mapped to one, and only one motion	D	One part may be mapped to one single motion, or the program will crash.	TRUE	Verification runs
	A motion from ADAMS/car can be mapped to multiple CATIA instances	D	Several CATIA instances may be mapped to the same motion as they may move as a system.	TRUE	Verification runs
	Fix a certain part in the CATIA assembly	W	In order to analyze movements relative to the fixed part.	TRUE	Verification runs
	The layout is intuitive and logical	W	With regard to work flow and the amount of information displayed	TRUE	Verification runs
	Mapping of a CATproduct includes all its CATparts	D	If not fulfilled the application will crash.	TRUE	Verification runs

Post-processing	The minimum distance between two CATIA instances can be measured throughout the replay motion	D	-	TRUE	Verification runs
	Measured minimum distance(s) can be exported to excel	W	-	TRUE	Verification runs
	Shortest distance between all CATIA instances can be measured throughout the replay motion	W	-	TRUE	Verification runs
	The distance measures are carried out within the replay file	W	-	TRUE	Verification Runs
Maintenance	Documentation of used variables in code	W	-	TRUE	Read through of code
	Code contains describing comments, where necessary	W	-	TRUE	Read through of code

D

An example of .res file

```
<?xml version="1.0" encoding="UTF-8"?>
<Results xmlns="http://www.mscsoftware.com/:xrf10">
<Bibliography>
<File URI="" schema="xrf" version="2.0.0.0" publicationDate="" />
<Corporation author="MSC.Software" URI="http://www.mscsoftware.com/" />
<Author user="" name="" />
<Revision version="1" derivedFrom="">
<Comment>
</Comment>
</Revision>
</Bibliography>
<Analysis name="" executionDate="" Solver="2013␣Adams␣C++␣Solver"  script="">
<Bibliography>
<Environment operatingSystem="" hostname="" />
<Application name="Adams/View" version="2013.00.00" />
</Bibliography>
<ModelInfo title="single_part" creationDate="-unknown-" checksum="-unknown-" />
<Units angle="deg" length="mm" mass="kg" time="sec" />
<StepMap name="map_001">
<Entity name="time">
<Component name="TIME" unitsValue="sec" id="1" />
</Entity>
<Entity name="Part␣1" entity="PART_2" entType="Part" objectId="2">
<Component name="X" unitsValue="mm" id="2" />
<Component name="Y" unitsValue="mm" id="3" />
<Component name="Z" unitsValue="mm" id="4" />
<Component name="PSI" unitsValue="deg" id="5" />
<Component name="THETA" unitsValue="deg" id="6" />
<Component name="PHI" unitsValue="deg" id="7" />
</Entity>
<Entity name="Motion␣2" entity="MOTION_1.motion_t2" entType="Motion" objectId="2">
<Component name="FX" unitsValue="newton" id="20" />
<Component name="FY" unitsValue="newton" id="21" />
<Component name="FZ" unitsValue="newton" id="22" />
<Component name="TX" unitsValue="newton-mm" id="23" />
<Component name="TY" unitsValue="newton-mm" id="24" />
<Component name="TZ" unitsValue="newton-mm" id="25" />
</Entity>
</StepMap>
<Data name="modelInput_001" id="1">
<Step type="input">
0
400 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
</Step>
</Data>
<Data name="initialConditions_001" id="2">
<Step type="initialConditions">
0
400 0 0 90 44.999999999999993 -90
0 0 0 0 0 0
0 0 0 0 0 0
0 8652.135792633906 0 0 0 0
```

```
</Step>
</Data>
<Data name="dynamic_001" id="3">
<Step type="dynamic">
1
400 0 0 90 44.999999999999993 -90
0 0 0 0 0 0
0 0 0 0 0 0
0 8652.135792633906 0 0 0 0
</Step>
<Step type="dynamic">
2
400 1.1532794797541026 1.1532794797541026 90 44.999999999999993 -90
0 32.521079053507442 32.521079053507407 0 0 0
0 571.6990343873133 571.69903438730455 0 0 0
0 9156.5300228793731 0 0 0 0
</Step>
</Data>
<TerminationStatus runTermObject="" runStatus="Successful" />
</Analysis>
</Results>
```