Lappeenranta University of Technology
Faculty of Technology
LUT Mechanical Engineering
Master's Thesis

Esa-Pekka Kaikko

DEVELOPMENT OF GENERIC SIMULATION MODEL FOR MOBILE WORKING MACHINES

Examiners: Prof. Jussi Sopanen
M. Sc. Simo Sinkko
Lappeenranta, 18th of February, 2015

ABSTRACT

Lappeenranta University of Technology
Faculty of Technology
LUT Mechanical Engineering
Mechanical Engineering

Esa-Pekka Kaikko

Development of Generic Simulation Model for Mobile Working Machines

Master's thesis

2015

77 pages, 28 figures and 2 appendices

Examiners: Prof. Jussi Sopanen
M.Sc. Simo Sinkko

Keywords: Simulation, Generic, Real-time Simulation, Hardware-in-the-loop, Tractor, Multibody systems.

This master's thesis has been done for Drive! –project in which a new electric motor solution for mobile working machines is developed. Generic simulation model will be used as marketing and development tool. It can be used to model a wide variety of different vehicles with and without electric motor and to show customer the difference between traditionally build vehicles and those with new electric motor solution. Customers can also use simulation model to research different solutions for their own vehicles.

At the start of the project it was decided that MeVEA software would be used as main simulation program and Simulink will only be used to simulate the operation of electrical components. Development of the generic model started with the research of these two software applications, simulation models which are made with them and how these simulation models can be build faster. Best results were used for building of generic simulation model.

Finished generic model can be used to produce new tractor models for real-time simulations in short notice. All information about model is collected to one datasheet which can be easily filled by the user. After datasheet is filled a script will automatically build new simulation model in seconds. At the moment generic model is capable of building simulation models for wide variety of different tractors but it can be easily altered for other vehicle types too which would also benefit greatly from electric drive solution. Those could be for example wheel loaders and harvesters.

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
Teknillinen tiedekunta
LUT Kone
Konetekniikka

Esa-Pekka Kaikko

Geneerisen ajoneuvosimulaatiomallin suunnittelu liikkuviin työkoneisiin

Diplomityö

2015

77 sivua, 28 kuvaa ja 2 liitettä

Tarkastajat: Professori Jussi Sopanen
DI Simo Sinkko

Hakusanat: Simulaatio, Geneerinen, Reaaliaikasimulointi, Järjestelmään kytketty komponentti, Traktori, Monikappaledynamiikka.

Tämä diplomityö on tehty Drive! –projektiin liittyen, jossa on kehitetty sähkömoottori liikkuvia työkoneita varten. Geneeristä simulointimallia tullaan käyttämään tuotteen markkinointiin ja tutkimukseen. Sillä pystytään mallintamaan monia erillaisia työkoneita, joko normaalilla moottorilla ja voimansiirrolla tai sitten uudella sähkömoottori ratkaisulla. Tällä tavalla voidaan paremmin näyttää asiakkaalle miten uusi tuote vaikuttaa ajoneuvon ominaisuuksiin. Asiakas voi myös käyttää simulointimallia omien ajoneuvojensa kehittämiseen.

Työn alusta asti oli päätetty, että simulointi tapahtuu pääasiassa MeVEA simulointi ohjelmistolla niin, että se toimii yhdessä Simulink ohjelman kanssa. Kaikkien elektronisten komponenttien simulointi tapahtuu Simulink ympäristössä. Geneerisen mallin kehittäminen alkoi näiden kahden ohjelman tutkimisella sekä tutkimalla niillä tehtyjä simulointimalleja. Etsittiin myös muita ohjelmia, jotka pystyvät toimimaan yhdessä MeVEA:n ja Simulinkin kanssa. Lupaavimpia tuloksia hyödyntäen valmistettiin geneerinen simulointimalli.

Valmis geneerinen malli on nopea ja sitä voidaan käyttää uusien simulaatiomallien tuottamiseen traktoreista. Kaikki tieto mallinnettavasta traktorista kerätään yhteen lomakkeeseen, jota on helppo muokata. Lomakkeen täyttämisen jälkeen automaattinen ohjelma rakentaa sekunneissa uuden simulointimallin lomakkeeseen annettujen arvojen mukaisesti. Tällä hetkellä geneeristä mallia voidaan käyttää vain erillaisten traktorien mallintamiseen, mutta sitä voidaan helposti laajentaa kattamaan myös toisenlaiset ajoneuvot, jotka hyötyvä elektronisesta voimansiirrosta. Näitä ovat esimerkiksi pyöräkuormaajat ja harvesterit.

ACKNOWLEDGEMENTS

This master's thesis has been done to Machine Dynamics Laboratory, at the Department of Mechanical Engineering, Lappeenranta University of Technology. The development of generic simulation model for vehicles is part of the Tekes funded DRIVE! –project which is done in cooperation of Saimaa University of Applied Sciences and Lappeenranta University of Technology.

I want to thank my examiners and supervisors Professor Jussi Sopanen and M.Sc. Simo Sinkko about the subject of the thesis, great support and good constructive feedback they gave me. I also appreciate help which I got from my coworker in Machine Dynamics Laboratory, Eerik Sikanen. Thanks to my fellow students and friends with whom I could exchange advices about the subject and sometimes next to it.

Finally, I want to express my gratitude to my parents who has been supporting me during my thesis and throughout my life. Also special thanks to Anniina who gave me strength and cheered me at home.

Lappeenranta, February 18th 2015

Esa-Pekka Kaikko

TABLE OF CONTENTS

ABBREVIATION LIST

UAS	University of Applied Sciences
LUT	Lappeenranta University of Technology
MeVEA	Mechatronics and Virtual Engineering Applications
HIL	Hardware-in-the-loop
2WD	Two-wheel drive
4WD	Four-wheel drive

1 INTRODUCTION

This master's thesis focuses on development of generic model for working machines. This model will be easily adjustable for different type or size of the same type of vehicles. Modelled vehicles will be used in simulation software MeVEA to simulate working cycle of working machines and its components in action. Simulation model is used to advertise actual components of simulated machines. These physical components are connected to simulator and they will move and work as simulated components do. This is good way to show customer how companies products will work and even in customers own vehicle. Generic simulation model can also be used to produce simulation models for product development. Fast modeling speed makes it possible to test new and different concepts in short notice.

1.1 Background

Society tightens rules relating traffic emissions and this brings new challenges for vehicles. Old solutions needs to be improved and new ones researched. All kind of electric vehicles and powertrains are developed and improved even further because they have their advantages when compared to traditional engines. Some of these advantages are that with electric motor it is possible to travel with cheaper price when it is compared to traditional engine, there is no gas emissions and price of the energy is not related to oil reserves. Electric vehicles can be categorized to different types such as hybrid electric vehicles, plug-in hybrid electric vehicles, battery powered electric vehicles and fuel cell hybrid electric vehicles. [1], [2]

One example of how to decrease the amount of traffic emissions is to use plug-in hybrid electric vehicle which is shown in Figure 1.1. It uses two or more different kind of power sources which can be for example combustion engine and electric motor. Both of these power sources are working together according to the orders of control unit. When vehicle is accelerating both of these power sources can work together which makes it possible to use smaller engine. When driving speed has been achieved engine can be used to keep the driving speed correct and to charge batteries with the help of electric motor for later usage. While car is idle or driving at

slow speed it is possible to use only electric motor while engine is dormant. Plug-in hybrid vehicles can be charged from external power source. [1], [2]

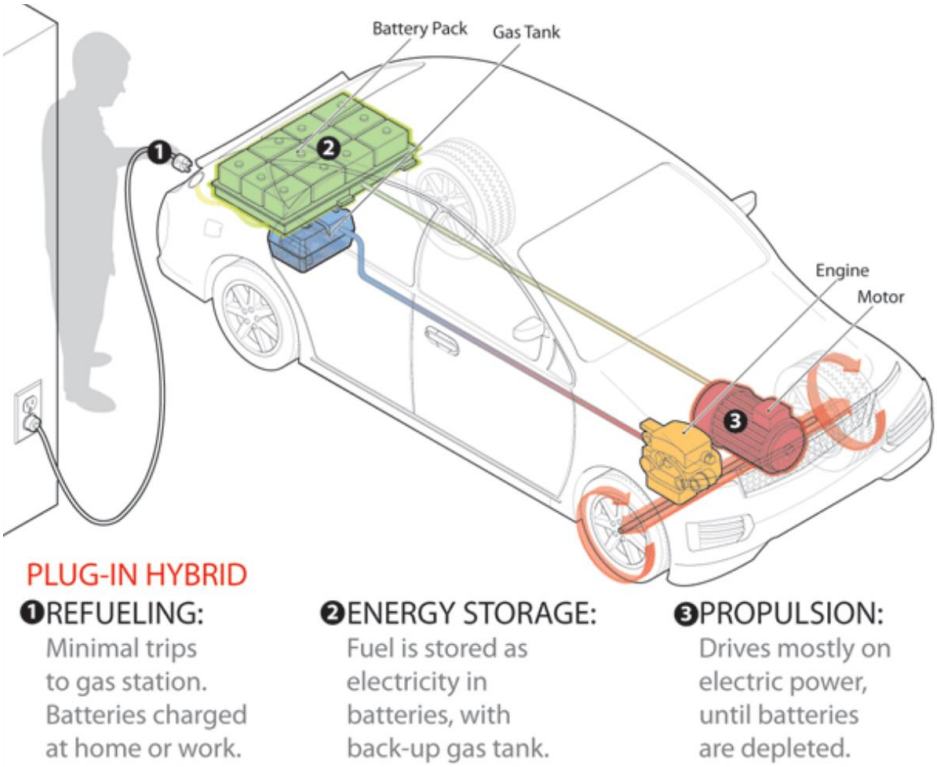


Figure 1.1 Plug-in hybrid electric vehicle. [3]

More strict rules about traffic emissions will cause problems especially for driven machines like tractors, wheel drivers, harvesters and other machines which are meant for heavy labor. For heavy vehicles like wheel loaders development has not gone very far and new technology is more than welcome. Electric driveline is not new invention but at the moment this technology is not as compact and efficient as it could be. Electric drivelines are big and they are not so efficient in work cycle which includes heavy labor and driving between destinations. When operator is driving to destination vehicle needs to move fast and while doing heavy labor vehicles transmission needs to produce high torque. To achieve this components, like electric motors which is shown in Figure 1.2, needs to be developed even further.

Figure 1.2 Electric motor which can be installed on every wheel of the vehicle. [4]

Saimaa University of Applied Sciences (Saimaa UAS) and Lappeenranta University of Technology (LUT) has worked together to produce compact electric motor for driven vehicles. This motor includes both motor and reduction gear which are combined to very small package. At the end of this project team intends to set up company which will develop electric transmission systems even further and sell them to customers. This company will also offer simulation services and also uses generic simulation model to assure customers that new electric motor solution will work and is really good option for customers own vehicles. Customer can drive a robust version of their own vehicle with simulator and see how electric transmission works in different situation in simulator. There is also real physical electric motor next to simulator (HIL, Hardware-in-the-Loop) which is connected to simulation model and customer can see from it how product works.

1.2 Aim and objectives

In this thesis a generic model for simulation software MeVEA will be developed. This generic model needs to be easily adjustable for vehicles of different sizes and types. With generic simulation model it is possible to make a robust simulation and show customer what is different

between traditional transmission solution and new compact electrical driveline with electric motor.

First objective is to make simulation model and environment for tractors. Models and environment at this point do not have to be graphically perfect and idea is only to show what new transmission can do. Simulation only needs to be so good that a difference between traditional and new electrical transmission can be noticed. A more accurate and complex simulation can be made if customer shows interest about product or for our own product development purposes. Aim is to make simulation model which will give a worthy demonstration of product and which will ease up customer with decision making.

After a working simulation environment and model for tractor is made it is possible to expand simulations to cover a variety of different vehicles. Same simulations can be done for example for wheel loaders, harvesters and other vehicles which would greatly benefit from low fuel consumption and driveline which can adapt to different situations during work cycle.

1.3 Applications

Generic model can be used to study electric driveline, to develop products or to sell them to customers. Model is easy and fast to be changed to describe different kind of products for different customers. It can also be used to analyze customers own problems and help them with product development. For example with simulation customers can try new different kind of steering and suspension systems which will become possible thanks to new compact electric driveline systems.

1.4 Methods and tools

Generic modeling process will start with studying of working vehicles. Different kind of vehicles from different manufacturers will be compared to each other to find similarities between products. These similarities will be used to make generic modelling process easier and much faster. Answers will be looked from literature and people will be interviewed to find most optimal way to build generic simulation model.

Real-time simulation software MeVEA has been chosen as platform for generic model. While simulations will be done mainly with MeVEA there are also several other software applications which are needed in generic modelling process. Modeling process will be divided in phases as shown in Figure 1.3. In first phase information about vehicle which will be simulated is collected from the customer. In second phase simulation model will be build according to the choices and after that simulation can be started.

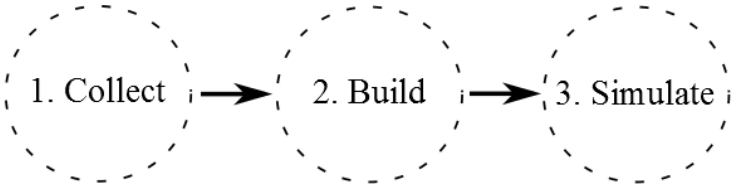


Figure 1.3 Working phases of building simulation model.

Gathering of information will be done so that it will be user friendly and also hasten building phase. It can be collected with paper questionnaire or it can be in electric form. Type of questions asked and information collecting should be done so that it will decrease the amount of time which is needed for modeling. MeVEA software will be studied to find out how simulation models can be build and what kind of answers needs to be collected from the customer.

At the building phase model will be build according to the information user gives at the first phase. Generic simulation model will be built from several sub-assemblies which can be replaced easily if it is necessary or their dimensions and other values can be changed. If simulation information has been collected in digital form there is possibility to make automatically updatable simulation in which programs will read answers from the datasheet and build simulation model according to the choices made.

Third phase is simulation phase in which product is ready and can be simulated. At simulation phase MeVEA simulation model needs to work together with Simulink which simulates working of the electrical components and also controls physical electric motor. This connection is must and it will be studied beforehand so that it is possible to make and it will work.

2 SIMULATIONS IN PRODUCT DEVELOPMENT AND MARKETING

Real-time simulations are good way to research and show behavior of various components or systems. This of course means that simulation model needs to be well build so that it is accurate and answers to all of customer's needs. A recipe to good generic simulation model is to do some research of the background which will guide model development in correct direction. Variety of information is needed like information about simulations overall, from mobile working machines and about different drivelines and from hardware- and human-in-the-loop systems. Good background research leads to good results and in this case to good generic simulation model.

2.1 Simulations

Simulation is one important method of investigation. Simulations can be used to gather information from expensive machines, to study mechanics and behavior of machines and to teach people. When system or problem becomes complex simulation is often most efficient method time- and moneywise to study problem. [5]

In simulation computers are used to evaluate models numerically. It is important to remember that evaluation is done with estimated physical conditions and simulation model cannot be as detailed and accurate as real system because it is not possible to describe all physical qualities and rules accurately. The other reason is that computers can only calculate a specific amount of information at the same time. Even if computers could handle gigantic simulation models programmers cannot describe all physical phenomena without simplifying them somehow. Most important thing time- and moneywise is to find balance between results and level of details. There is no need to improve accuracy of the simulation too much because at one point results will not improve as much as modelling takes time. At the start of making the simulation model modeler needs to know what kind of simulation model is accurate enough. It is better to start by making a little inaccurate model at the start and improve accuracy later. [5] By doing simulation model this way it is possible to get initial results sooner. Another advantage of this method is

that troubleshooting is simpler because the amount amount of processed data and simulation model is much smaller.

Simulations can also be used for user training. This is really effective when people are learning to use expensive or dangerous machinery because simulations can be recorded and work cycle and behavior of the laborer and machine can be studied afterwards. It is possible to teach workers to use heavy machinery without a risk of accidents or human casualties. One really good example is flight simulation of passenger planes. Pilots can study flying with commercial planes on a safe environment without a need of endangering expensive plane and lives of hundreds of passengers. Also simulations can be easily altered for emergencies and pilots can study special situations like landing on ocean or flying with lowered engine power.

There can be different kind of simulation. In continuous simulation some variables will change with respect to the time. For example some differential equations can give rates of change for some variables like flow of water or speed of the car as shown in Figure 2.1. Another kind of simulation is discrete-event simulation in which state variables will change instantaneously. This instantaneous change can for example mean changing the gear of the simulated car which is also shown in Figure 2.1. In many simulations it is not possible to say if simulation is either continuous or discrete but it is a combination of both. [5], [6]

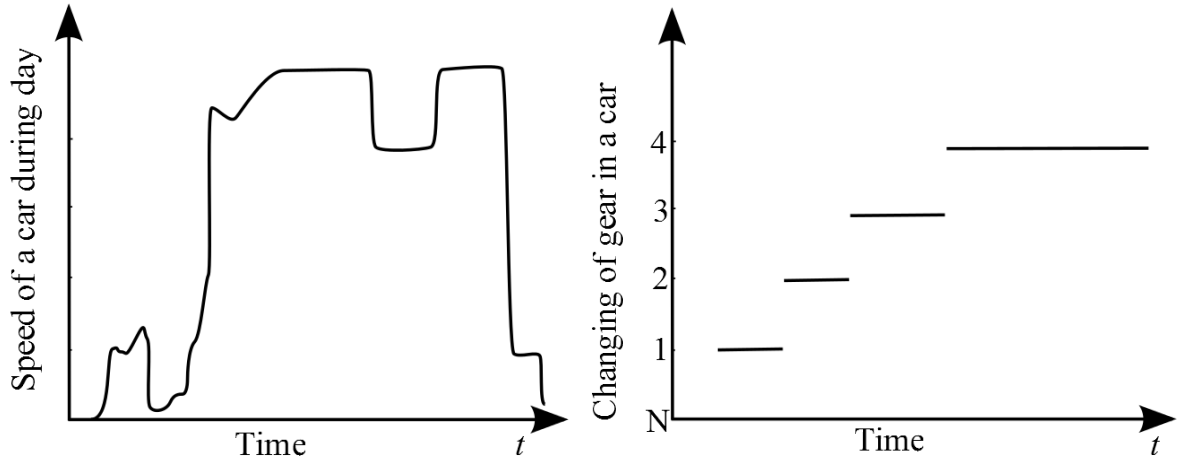


Figure 2.1 Continuous and discrete-event simulation.

2.1.1 Advantages of simulation

Simulation can be really good research tool when used for correct purposes but it can also be big failure time- and moneywise. One big advantage of simulations is that it can be used to study behavior of complex systems. When system under investigation becomes more complicated it is harder to describe it with mathematical model which is made analytically. [5] Also studying behavior of physical system can be done with simulation without a need to disrupt work of real machine. For example this will save a lots of money when dealing with big factories. [6]

Simulation is good tool when different systems needs to be compared to each other. When first simulation model of the system is done it can be easily altered and testing of different variations of same system is cheap. This is good way to find out solution which best meets demands of customers. Maintaining specific experimental conditions is also much easier with simulation than in real life. [5]

One of the biggest advantages of simulations is changeable time frame. Tests which are done with physical products can take years to perform but there will be results right away when simulation is used. In simulation it is also possible to do test in small time frame and examine system or process in small detail. [5]

2.1.2 Disadvantages of simulation

Simulations will not come without drawbacks which are good to remember. Simulation only gives estimate characteristics of the model with specific variable values. These results only shows a small portion from whole answer because some variables are estimated or they would keep changing in physical system. To get reliable information about system a running of several simulations with different variables is advisable. [5]

While simulation offers a change to study system without a need to manufacture a real physical machine they can still consume much time and money. [6] Simulation is not good for all purposes but it will shine when used for continuous research or development of same kind of system. Making of simulation model for single usage is not always profitable because making

of first simulation model takes a lot of resources. But when it is ready it can be easily modified and used for simulation of different variations of same system. [5]

When simulation is ready and running it gives a great volume of numbers, values and other kind of data. Impressive amount of simulation data can overwhelm researcher and it can easily be used with greater confident than which is justified. Results from simulations should be questioned just like information from all other sources because impressive results from simulation will not mean that simulation has been done correctly. [5]

2.2 Multibody dynamics in simulations

Multibody dynamics is important tool for design and simulation of complex mechanical systems. In multibody system actual system is replaced by equivalent model which is made from discrete bodies like the one in Figure 2.2. These bodies can be rigid or deformable, they can undergo rotational and translational displacements and they have their own elastic and inertia properties. [7], [8]

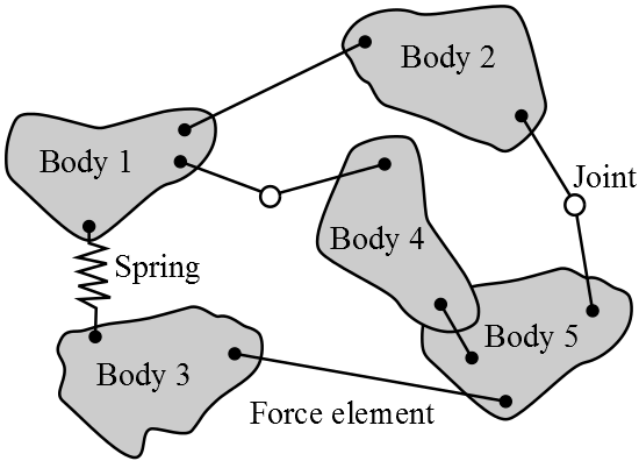


Figure 2.2 Multibody system. (Modified [7])

One from many software applications which uses multibody dynamics in its calculations is MeVEA. MeVEA simulation software uses global and local coordinate system which is easy method to define location and orientation of parts in multibody systems. Multibody system

needs one coordinate systems which will be used for calculation purposes and objects location and orientation can be defined with it. Every object also has their own coordinate system which is called local coordinate system. While objects location and orientation can be described with global coordinates they can also be defined with help of local coordinates of other objects. [9]

After objects are placed to the simulation environment some constrains between parts needs to be made. Different bodies can be connected to each other in different ways depending on the simulated system. A couple of example constrains used between different parts in MeVEA are hinge, spherical, universal, translational and cylindrical joints. Parts can also be fixed to each other or on to the same plane. It is important to pay attentions so that correct constrains are used because they will define how movement of one body will affect behavior of whole system. [9]

Constrains will define how different bodies are connected to each other and how they will affect whole system. Behavior of whole system depends on masses of single bodies and forces which affects on each part. To simplify calculations masses of every part are thought as particle masses. Every object has one particle mass in it in specific location inside the body. Inertia values are also defined for every part. Inertia tells how weight of the body affects on objects movement. [9]

2.3 Simulators

Simulators include hardware and software which are working together to produce as realistic and accurate information about simulated object. Simulators are built so that they can give user as realistic feedback as possible with the help of sound, touch and vision. Some simulators can even use taste and smell to tell user various things.

In this master's thesis a simulator which is shown in Figure 2.3 will be used. It includes two different computers which controls Simulink and MeVEA programs and one motion platform. User will sit in the motion platform and control simulatable vehicle from there. In this project there will be also electric motor next to the simulator of Figure 2.3 which moves according to the simulation.

Figure 2.3 Simulation platform in Lappeenranta University of Technology.

Most simulators will use one or more screens to give visual feedback to the user. One example could be video game consoles which uses television or other screens to show what happens in simulation. Video games will also use sense of touch to relay information to user. Most controllers which are used to play game consoles uses vibration and will shake in gamer's hand. Bigger and more expensive simulators can move and shake whole body of the user. MeVEA is one of the companies which offers simulations and simulators for people. Their advanced simulators can even have many different screens to inform user visually, motion platform to simulate shaking and moving of the simulated vehicles and all impacts it makes and of course speakers to give sounds for various operations. [10]

Simulation can relay information for user in many ways but it also works in another way too. User can control simulation with various controllers and sometimes even with sound and gestures. Depending on the simulator it is also possible to use real chair and controls of the simulated vehicle. Using of the real controls of the machine is useful especially in user training. That will make user training safe and there is no need to risk expensive machinery to the hands of newly recruited worker.

2.4 Hardware-in-the-loop and human-in-the-loop

Simulations can be used alone but there is also a possibility to add physical components to it. In that case simulation would be called hardware-in-the-loop (HIL) simulation. One good way to use HIL simulations is testing and product development as Ford Motor Company does. [11] One example for HIL testing applications is testing of brakes of the cars. In this example a car will be simulated in virtual environment and hardware (brakes) are connected to test bench. When car is driven around virtual environment simulation will tell brakes what to do. When brakes are used in simulation environment a physical component will brake too.

There are many advantages if HIL is used. Noise factors can be eliminated or at least controlled, development times are shorter and this method saves money. Testing hardware and area is small when compared to testing of a whole object and possibilities are much wider. Same simulation with only minor changes can be used for testing of different parts. Also amount of prototypes will be decreased. [11]

Human-in-the-loop works same way as hardware-in-the-loop. Now a driving simulator is used as an example. A human will drive a vehicle in simulated environment and communication between simulator and human is done with screens, pedals and steering wheel. Simulator will get data from pedals and steering wheel and runs simulation according to the information it gets. At the simple systems driver gets feedback through screens but when system gets more advanced there are much more possibilities for simulation to communicate with driver. For example it is possible to give feedback to the driver with motion platforms, shaking steering wheel or sound.

Simulation uses most of these methods to communicate with driver in example of human-in-the-loop system of Figure 2.4.

Figure 2.4 Human-in-the-loop simulation. [12]

There is also a possibility to combine different kind of in-the-loop simulations together. For example a person can use driving simulator and observe while hardware-in-the-loop works in real time according to the simulation.

2.5 Marketing products with the help of simulation model

Simulation models can also be used on selling purposes. When companies will buy something bigger they rarely see those products beforehand. It is also more unlikely that they can test them beforehand. Simulation model makes it possible to advertise product by showing how it works in different situation and maybe even with customers product. For example a good way is to show a difference in customer's product with advertised item and without it. If customer can be assured about capabilities of advertised product they will more likely buy it.

Another possibility is to sell simulation itself because many companies cannot afford a simulation machinery of their own or they do not want to expand their area of expertise to simulations. In those cases companies needs to be informed about the possibilities of simulations in research or design. Customers can buy simulations which will help them with research and save them money and time. If simulations is done accurately it can be used to pinpoint flaws and faults in products and fix them before building machine itself. It is cheap way to test working of different machines and systems if you compare it to testing with physical machine. Faulty components can break whole machine or even risk lives of workers.

3 GENERIC SIMULATION MODEL

Modelling processes are not always so different and when modeling is focused on specific items more similarities can be found between products. Finding parts which will be modeled generally in same way will decrease the amount of time that is dedicated to model a new machine. Modeling same kind of parts and making same kind of simulations will waste valuable time which could be used for more important tasks. Similarities between different products can be used to construct generic model which will greatly decrease the amount of time that is needed to model new items.

3.1 Basic working principle of the generic model

Main purpose of generic model is to hasten modeling process of same kind of products. While making of first model will take lots of time each model after that will be modeled really fast. When the amount of modeled products increases generic model will become even more useful. It will also save a lot of money and time which can be afterwards spent on other targets.

Figure 3.1 shows basic idea of different working phases in generic modeling. Everything starts from collecting information about the product. What kind of product will be modeled is decided now on phase 1. How to put these information in use varies depending on the generic model.

There is many possible solutions and one is to insert model information straight to the simulation software and model. Simulation model could use some kind of database or item library which includes several different products or their sub-assemblies. According to the choices user will make simulation model will be built from these blocks.

Another possible method to collect model information is to make a datasheet in which all necessary values and questions about simulation model will be asked. Simulation user will fill datasheet which will be read afterwards by some program. This program will then automatically build simulation model according to the choices user made to the datasheet.

The core of the generic model is the second phase in the Figure 3.1. In phase two simulation model will be generated according to the choices user made at phase one. There is many different options of how simulation model is made as generic and how model will be built after all necessary information is gathered. Which method is the most efficient and useful highly depends on simulation software which is used. From all solutions two examples are now demonstrated.

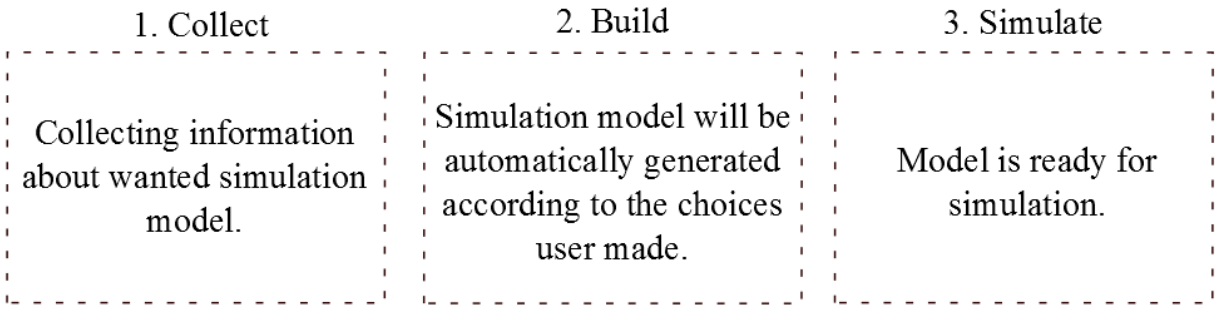


Figure 3.1 Basic idea of working phases in generic model.

First option is to use simulation software itself to build simulation model. This method can be made generic for example by dividing simulation model to small pieces and modeling them beforehand. For all of these sub-assemblies there are several alternative models which can be used to form new simulation model. Although making a library from several parts will take some time it will pay itself back when more and more parts are made.

Because simulation model can be assembled from many smaller parts there are infinite different assembly possibilities. When even one part is changed to another it can make huge impact on the behavior of simulation model and make it completely different. For example only by changing engine power or transmission it is possible to make a huge difference between two different simulation models.

Figure 3.2 shows an example of assembling simulation model from smaller components and sub-assemblies. In Figure 3.2 model is divided in to five sub-assemblies which are frame,

transmission, test environment, wheels and engine. For each of these sub-assemblies there are several different models which can be chosen. For wheels there are six different models which can be used. In addition to sub-assemblies there can also be other variables like masses and inertias which can be changed.

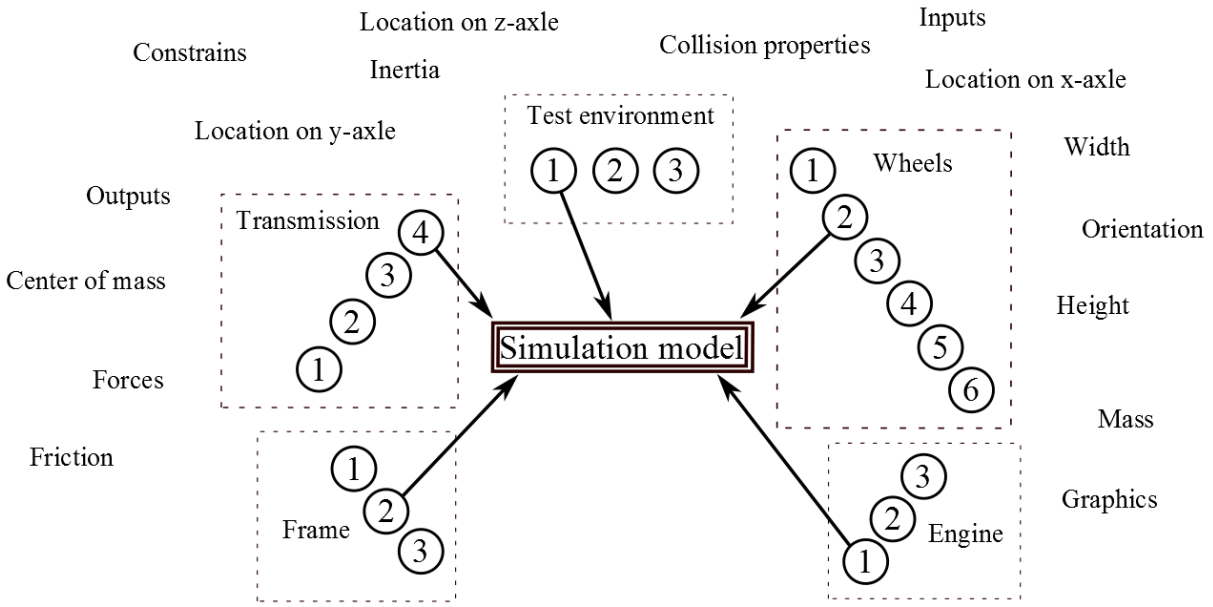


Figure 3.2 Building simulation model from different sub-assemblies.

In another example main focus will be on simulation files. For now simulation software can be forgotten and focus is entirely on simulation files. Different simulation software applications will save simulation models in different file formats and in some cases those files can also be easily opened and modified with other programs. When model files are ready simulation program itself is needed only to run the simulation model at phase three.

Depending on which software is used simulation files can be saved in file formats which can contain easily understandable coding and text. If possible coding of these files can be made easier by dividing all information inside them to smaller packages by what is their function. For example all text and coding which are related to the transmission should be divided to entirety of its own to make further modification of model easier. As demonstrated in Figure 3.2 for every sub-assembly there could be several different possible layouts which can be used. At this phase

sub-assemblies can be a little plain because they only need to contain basic structure of the component.

When simulation files are being built sub-assemblies can be used to speed up modeling process. Every sub-assembly will have their own coding packages which will be written on simulation files according to the choices user makes. When simulation files are assembled and a basic structure of the model is ready it is possible to do some fine tuning. For example at this point it is possible to adjust properties of model like mass of different parts, their inertias, locations, graphics, height and width.

To make model generic there needs to be fast method to modify simulation files. There is many different possibilities to make writing of simulation model files easier and faster. One of these methods is to write a script which will do specific modifications depending on the orders it has been given. For example if user wants that simulated vehicle has four-wheel drive script will build simulation file according to that choice.

3.2 Identifying common elements between mobile working machines

There are different kind of working machines because every job needs different kind of vehicle and every manufacturer builds their own variation of machines. These working machines will sometimes diverse a little or significantly. While different machines has components of their own there is still some common elements between vehicles. Although example vehicles in Figure 3.3 are different type and from different manufacturers it is possible to point out some common elements. Now those common elements, which will have biggest effect on the results of the simulation of vehicle, will be defined.

Figure 3.3 Many similarities can be found even between different type of working machines from different manufacturers. [13], [14]

First common elements between working machines are energy storage and motor. Fuel tank and batteries can be considered as an energy storage which will give energy when it is needed. Motors and engine can use that energy and change it suitable for different situation. It can be used to give electricity for lights, radio and hydraulic pumps or it can be used to produce torque to axles, wheels and shafts.

Hydraulics can be considered as one set and it can be found from many working machines because it is most efficient method to produce force for the jobs. It is used to control different kind of arbitrary equipments like frontal and rear bucket. It is also used to control hitch on tractors.

Working vehicles needs to produce huge torques so that they will perform well in harsh environment while doing heavy labor. Engine cannot be connected straight to the wheels because then it would not produce as much torque as is needed. This problem can be overcome with reduction gears and gearbox which will decrease rotation speed of the wheels while increasing the amount of torque transferred to them.

Wheels, and in some cases crawler tracks, can be thought of as one common element between working vehicles. They can change in type, size and weight depending on the size and type of the machine. Even the amount of wheels can change depending on the vehicle and the job it

does. For example when working on swamp with a tractor driver can equip vehicle with extra wheels to distribute weight of the vehicle to bigger area of the land and prevent it from sinking in the swamp.

While inside components of the vehicles are important the outer case and cabin are as essential. Cabin and outer case of the vehicle is there to protect equipments and driver from weather and mud which will increase operating time of the machine. Driver will be able to use machine longer and vehicle will not break so often.

Excess working equipments like trailers can be considered as an individual set because all working vehicles needs to be able to do many kinds of work. These excess labor equipments can for example mean trailers, different kind of farming plows, buckets or maybe drills. Additional labor equipments are only extra to the vehicle and vehicle is fully function even without them.

3.3 Dividing model to sub-areas

To ease up understanding of the simulation model and to make modelling process easier simulation model will be divided to sub-areas which can be thought as entireties of their own. Advantage of this method is that later on model can be changed easily and different parts can be found faster. Figure 3.4 shows an example of how wheel loader can be divided to sub-areas.

Figure 3.4 Generic simulation model divided to several sub-assemblies. (Modified [14], [15], [16], [17])

First sub-area will be environment which can be easily changed to other kind of environment models depending on which kind of work will be simulated. There is no need to add all possible work cases in one huge simulation which will be hard to handle and use. In that way it is possible to decrease the amount of computational power which is needed. Depending on the wanted work case environment can be changed to different ones. For example loading simulation can take a place in sandpit, driving simulation in bumpy roads and logging simulation in forest. Environment will include day cycle, sounds and some movable object like logs and boulders which can be included to the simulation.

Cabin and outer case of the vehicle will protect the driver, engine and other parts from weather while they also improve appearance of the vehicle. These two can be thought as one entirety which will only affect simulation by adding mass on the simulation model.

Motor, gearbox and transmission can be considered as one pick package because they are so tightly connected to each other. Another reason for thinking these as one entirety is MeVEA software and the way how these components and their relations needs to be modeled in it.

Wheels are on the other end of the transmission but still they are divided as a group of their own. Reason to this is the amount of information and modeling which will be needed when wheels are modeled. While components like gearbox do not need to be visualized and can be described only with a couple of lines of codes, wheels have a strict modeling rules of their own. Main interaction between vehicle and world will be done with wheels.

Hydraulic system can be easily separated as its own group. While it is commonly used for controlling of the additional labor equipments and hitch it will only have a minor role in simulation at the beginning. Later on, when simulation model will be improved and other addition labor equipments will be added, hydraulic system can be improved.

Last sub-area for simulation model comes from additional labor equipments which includes plows, trailers, buckets, harvesters and extra frontal or rear weights. This simulation model group can be easily increased with different kind of labor equipment.

3.4 Simplifying parts and elements

Before simplifying parts it is better to know the working principle of multibody simulation software applications like MeVEA. When parts are modeled in MeVEA at the start they do not have any graphics and shape and they cannot collide with other objects or simulation world itself. These parts can then be given visualization and collision graphics if needed. Visualization graphics will make object visible but will not add anything else to the part. Collision graphics adds collision surfaces for the object and after that part can collide with other parts with collision graphics of their own. Collision can also be defined with splines which is one method used to describe collision of the wheels. [9]

Parts do not have any shape and they will be placed in some locations on the simulation model. With constrains and relations to other parts every part will be placed one at a time to simulation environment. Every part needs to be placed to the simulation in relation to the coordinate system. It will not matter which coordinate system is used. Parts can also be connected to each other from any place because they have no shape.

First simplification of the generic simulation model is the frame which will be used as building platform for other components. This frame will not have shape, visualization graphics and it cannot collide with anything. It is only used as modeling platform and to connect different parts together. Reason for using this building platform is to decrease the amount of connection chains between different parts and to make modeling faster and easier. When parts are connected to each other these connections are defined between parts with specific name. If some parts which are located somewhere in the middle of connection chain are needed to be changed to different kind it cannot be done so easily. When parts are connected to the frame which will be used as common building platform changing of one part is easy and can be done quickly.

Some other parts also will be modeled without visualization graphics. These parts only have a mass which will be connected to correct location on the frame. For example battery packages, motors, generator and fuel tank will be modeled like this. Working of components will then be described with lines of codes and rules. If engine is taken as an example it will be added to simulation model as mass and working with other components like transmission and gearbox will be described by using codes, splines and so on. These parts will also be modeled without collision graphics.

Outer core of the vehicle and cabin will be modeled with visualization graphics. They also have mass and inertia properties, but they cannot collide with other object. Wheels are modeled with visualization graphics and mass and inertia properties but in comparison to previous parts wheels can also collide with objects. Collision properties of the wheels are modeled with splines which will define profile of the wheels. Figure 3.5 shows an example in which wheels profile is defined with 3 splines. Here center of the wheel has biggest radius while each side of the wheel has smaller one. [9], [18]

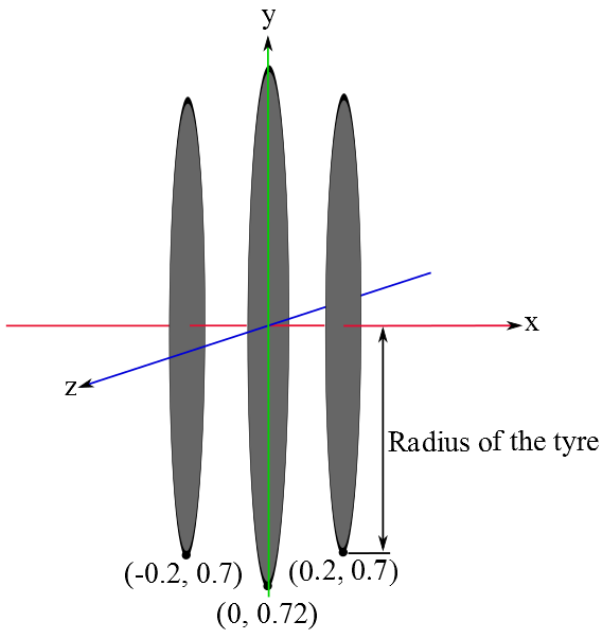


Figure 3.5 Tyre profile described with three spline points. (Modified [18])

Transmission will be modeled only by adding relations between different parts and several masses to the corresponding locations of model. Gearbox, transmission, motors and wheels will be related to each other and work regarding to commands of the driver of the simulator.

In this simulation a simplified environment model of the MeVEA tutorial 3 will be used. Tutorial environment will be simplified by removing all fences, dumpsters and other objects from it. There will only remain three objects on the environment model and those are ground, sky and tree line. Ground is flat and it has only simple visualization graphics. It is the only object in simulation model which can collide with wheels. Over the ground is huge dome of which inner surface will mimic the sky. Simulation environment will be surrounded by a simple image of tree line. [19]

3.5 User environment

User environment should be as simple and easy to use as possible but it will only get little attention because it is not main goal in this study. User friendly choices are made when it is possible and when it will not limit the main goal of this thesis, making of generic simulation model.

At the start of this thesis it was known that at least three different programs was needed to make the generic model work. All of these programs will need information about simulated product and all of these programs needs to be updated with new values every time changes needs to be made to simulation model. To make changing of the generic model easier in user's perspective it would be wise to use some kind of common information datasheet. All necessary information about generic model will be collected in one place from where the model can be altered. All other programs should then read information from this common datasheet to perform their tasks according to the information. This will make further modeling even easier because altering of the whole generic model will be done from one place.

Depending on the type and structure of the simulation model datasheet will be built as simple as possible. When building datasheet and deciding how information is put there it is good to think who the user will be. Will it be someone from the working group or is it customer who will see that datasheet first time in his/her life. If datasheet is meant only for own working group then it can be a little rough. There can be all kinds of notes and every value do not have to be explained. It is really different story if datasheet will be used by customer because then it has to be clear, simple and there should not be any possibilities for it to be used wrong.

First it should be clearly built so that all the necessary information can be found easily. Second important thing is about how all necessary information is asked. For example if you can decide total length of the vehicle from datasheet it is important to make clear in what units answer should be given. Also a range of answer could be limited so that user will not choose too small or big values which could cause simulation to malfunction. A good way to avoid unwanted

answers is to use dropdown menus from where user can choose what properties he wants to include to simulation model.

In cases where simulation model will be improved all the time it is important to pay attention to user environment for yourself and co-workers. Previously mentioned datasheet can only be surface for whole project and it can only be used to change simulation model in some extent. For example if there is a need to make new tyre type for simulated vehicle it cannot be done with datasheet and some deeper modeling and programming is needed. In order to make further improving of the simulation model easier for other workers all model files and information needs to be archived and built clearly. Small notes and guides will help others and yourself to understand how a specific parts from the whole generic model works.

When all of previous things are summed up we get really simple principle for user environment. User environment should be as clear and simple as possible without risking the quality of the product.

3.6 Demands for the generic model

There is some demands for generic simulation model which are needed to be fulfilled. Most important demand is that modeling process needs to be much faster with generic model than without it. Simulation will be mainly used for marketing purposes so that some vehicle will be simulated and customer can test it with simulator. Goal is that customer will give information about their own vehicle which will then be simulated with generic model. It is possible to ask all necessary information about customer's vehicle beforehand but modeling process takes still too long. This is why it is important to decrease the amount of time which is needed for modeling. It should not take more than a day to make new simulation model according to customer's specifications.

Because simulation model will be built fast it can be graphically a little plain but it still needs to accurately simulate working of the real vehicle. Because customer will be put in to the simulator to test it there is no room for mistakes. If customers own product will be simulated

then he knows right away if simulation feels accurate or not. After testing simulated vehicle by doing its daily working cycle customer can confirm accuracy of the simulation. After testing their own vehicle, new marketed product can be attached to the simulation model so that customer can test it and compare it to old product. If customer thinks that their vehicle has been simulated accurately, they are also more likely to trust testing data of new product. This will make further conversation and marketing much easier because customer also gets better feeling about the product he is going to buy.

Generic model needs to be designed so that there can be many different kind of drivetrain option. This feature will make testing and comparing of new and old product possible. Some simulated vehicles will be modeled with combustion engines and some with electric motors and drivetrains. Generic model needs to be designed so that both of these totally different drivetrain options are possible.

Software applications which will be used for simulation has been decided beforehand and new programs needs to be able to work together with these two. Main simulation software will be MeVEA which will be supported by Simulink when electric motors are simulated. Generic model needs to be designed so that sometimes Simulink is connected to MeVEA and sometimes it is not. While Simulink will communicate with MeVEA it will also run real physical electric motor which is brought next to simulator. This electric motor will work according to simulation and improve reliability of real time simulation.

At first generic model needs to be designed so that it can be used to simulate different versions of one specific type of working vehicles. This is just to decrease the amount of work at the start and to get one type of vehicle working. Generic simulation model should be designed so that in the future it can also be used to model other types of vehicles.

While basic structure between many vehicles is quite similar there can also be huge differences between parts. For example sometimes size, type, location and mass of every part can vary significantly depending on the product and manufacturer. Also some parts can be used in one

vehicle while they are not used in another one. A couple of important dimensions are shown in Figure 3.6.

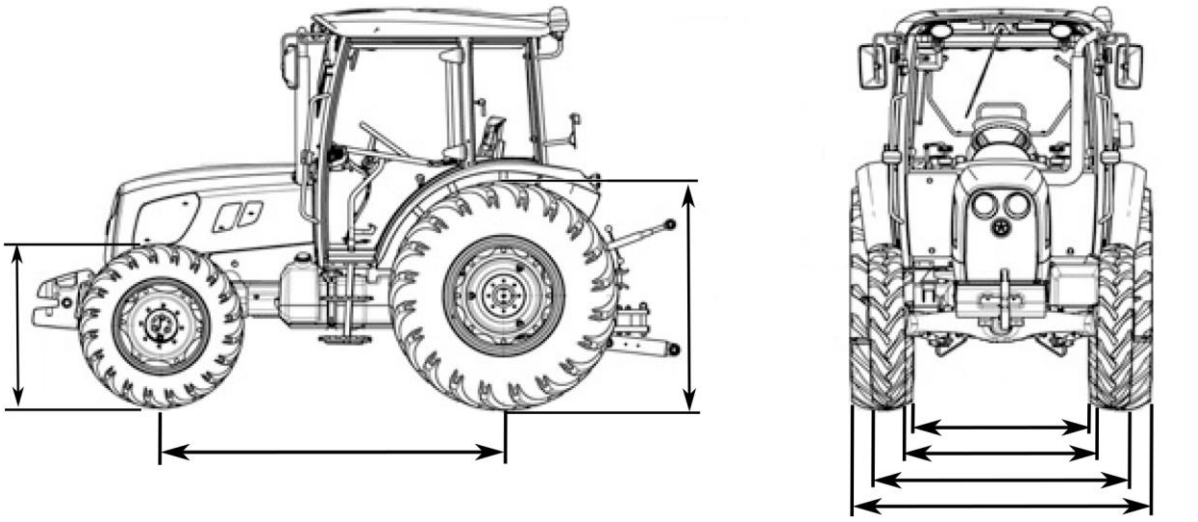


Figure 3.6 Few important dimensions for wheels which have huge effect on the driving behaviour of the tractor. (Modified [20])

For every vehicle there are dimensions and values which are more important than others and these should be easily changeable. A couple of these are masses, wheel sizes and their locations and inertia properties of different parts. Here is the list of properties which needs to be changeable easily.

- Dimensions, mass and inertia of the wheels
- Distance between front and rear axle
- Distance between wheels of the right and left side.
- Mass, location and inertia of the cabin and other structural parts of the vehicle.
- Engine's location and inertia
- Generator's location and inertia
- Location and inertia for 4 electric motors
- Fuel tank's location, mass and inertia
- Location, mass and inertia for battery package

- Efficiency of transmission
- Amount of work load
- Amount of the driven wheels
- Amount of electric motors used

All the values mentioned in previous list will be changed depending on the customer and that is why they need to be easily modifiable. Most of the parts needs changeable mass and inertia values and their location can change while a couple of parts will be completely removed in some cases.

Models which will be made with generic model can be really different. Maybe biggest differences can be found between vehicles with combustion engine and those which are driven only by electric motors. If conventional vehicle model would be upgraded to electrical one then combustion engine, mechanical transmission and fueltank will be removed and they will be replaced by battery package and four electric motors which are located in every wheel. Generic simulation model needs to be built so that modeling of the hybrid vehicle is also possible. In that case combustion engine will produce torque which will be transformed to electricity with generator. Electricity can be used to run electric motors, to charge batteries or to give power for various equipments.

3.7 Choosing software applications

For this simulation project MeVEA software was a natural choice because of the hardware and knowledge LUT and Saimaa UAS had from it. But because it was not possible to do everything with MeVEA software alone some other programs were also needed. Ideas of generic model, user friendly interfaces and hardware-in-the-loop systems also increased the amount of programs which were needed. All software applications and how they are connected to each other is shown in Figure 3.7.

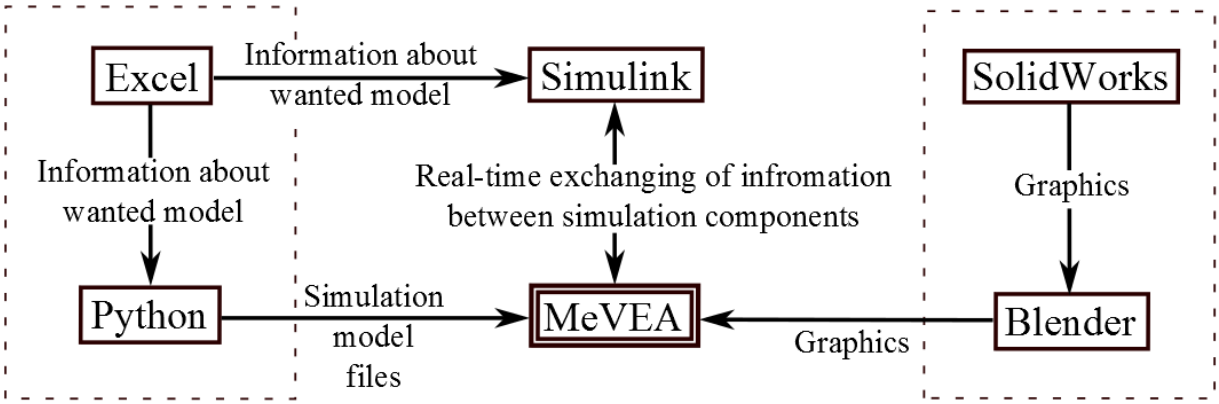


Figure 3.7 Data exchange between different software applications.

MeVEA software was chosen first as simulation software and next one was SolidWorks. SolidWorks will be used to produce graphics for simulation and it also gives inertias and masses for every object. While SolidWorks can be used to make accurate 3D models it cannot save them in correct file format which is used by MeVEA. This is why a free Blender software has been chosen as file format converter between SolidWorks and MeVEA.

Some coding was needed to be made to make simulation model generic. In this case Python stood out because of previous knowledge that it could do the job. While Python script builds simulation files it is not so user friendly and some other program was needed to gather up all the choices which user wants to make and what content he wants to add to simulation. A commonly used Excel was chosen for this job and it will be used as an answer sheet which will be read by python and Simulink.

Matlab's Simulink will get data from excel and it will communicate with MeVEA in real-time. While simulation is going on it will also work with hardware which is connected to simulation. Simulink model is not included to this thesis.

3.7.1 MeVEA

MeVEA is dynamic simulation program which is used for real-time simulations. Harvester and forwarder which are shown in Figure 3.8 are only a two examples from all the simulation models which are made with this software. MeVEA software has been developed by MeVEA

Simulation Solutions which have been founded in Lappeenranta. This company offers research and product development simulators, modeling and simulation services and simulation hardware for their customers. [10]

Figure 3.8 MeVEA harvester and forwarder simulators. [10]

In this project MeVEA software has been chosen as simulation platform because LUT and Saimaa UAS has some hardware and previous experience with it. LUTs MeVEA simulation platform which is shown in Figure 2.3 also made MeVEA software as natural choice.

MeVEA software will be used for real time simulation of heavy working vehicles and it is used on LUT motion platform. While MeVEA does most of the simulations some components and their behavior will be simulated with Simulink which will be working together with MeVEA.

3.7.2 SolidWorks

SolidWorks is widely used 3D modeling and simulation software. It can be effectively used to produce 3D models and 2D manufacturing drawings from machines and parts. It can be used for example to make mold, sheet metal, piping, tubing, plastic and cast part designs. [21]

SolidWorks has been chosen as graphic program because many of the possible future clients uses SolidWorks and they have modeled their products with it. By using clients own 3D models to make graphics for generic model it is possible to save lots of work. If SolidWorks models are done well it is also possible to use them to get masses and inertia values for simulation. Figure 3.9 shows an example of what kind of material properties SolidWorks can give. There is also a possibility to simplify model and decrease the amount of parts which will be used in generic model. By doing that simulation model will not be so heavy and it will run smoother.

```
Mass = 551554.33 grams

Volume = 551554330.37  cubic millimeters

Surface area = 18614540.90  square millimeters

Center of mass: ( millimeters )
    X = -1334.97
    Y = 659.47
    Z = -451.70

Principal axes of inertia and principal moments of inertia: ( grams * square millimeters )
Taken at the center of mass.
     Ix = (1.00, -0.00, 0.00)     Px = 179475981999.90
     Iy = (0.00, 0.87, 0.50)      Py = 429431383581.31
     Iz = (-0.00, -0.50, 0.87)    Pz = 508696046026.07

Moments of inertia: ( grams * square millimeters )
Taken at the center of mass and aligned with the output coordinate system.
    Lxx = 179475982209.71     Lxy = -7014667.47          Lxz = 3778997.47
    Lyx = -7014667.47         Lyy = 449120643616.22     Lyz = 34249023215.10
    Lzx = 3778997.47          Lzy = 34249023215.10      Lzz = 489006785781.35

Moments of inertia: ( grams * square millimeters )
Taken at the output coordinate system.
    Ixx = 531878359501.61     Ixy = -485577884401.11    Ixz = 332591408390.96
    Iyx = -485577884401.11    Iyy = 1544603072246.32    Iyz = -130047239862.26
    Izx = 332591408390.96     Izy = -130047239862.26    Izz = 1711824941040.88
```

Figure 3.9 Mass and inertia properties of example wheel loaders bucket.

3.7.3 Blender

Blender Foundation is developing Blender which is free open source software. Blender has huge amount of users worldwide. This software is advertised for example for its fast modeling speed, photorealistic rendering and amazing simulations. Blender Foundation also gave some examples of what Blender is capable of and made short movies. [22]

Because SolidWorks cannot save graphic files to compatible format with MeVEA some other program is needed for converting. Blender has been chosen mainly because it is free and fairly easy to use. It can also be used to paint parts and to add some logos and writing to them. Blender will be used to open SolidWorks .wrl files and to save them as .3ds files which can be used by MeVEA.

3.7.4 Python

To make simulation model generic it was needed to write scrip which would write new simulation files for MeVEA automatically. Python software and code was chosen to this because it is fairly simple to use and working group had some previous experience from that programming language. The advantages of Python is that it is versatile and widely used open-source programming language which can be learned and used easily. It is constantly improved by Python Software Foundation which will also promote and protect it. [23]

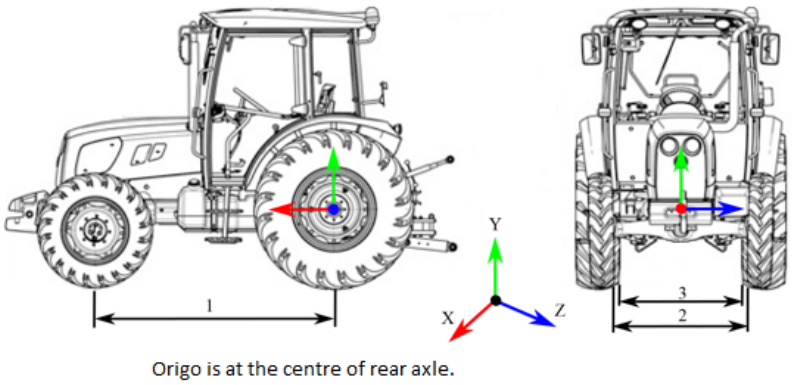
In this master's thesis python programming language is used to make a script which will build .xml file and several .mva files which will be later used in real-time simulation. Before making of new files Python script will read Excel-datasheet. After reading datasheet script will build simulation files according to the choices which were made to Excel-file.

3.7.5 Excel

Simulink and Python can look difficult and they are not so easy to use. That is why Excel was chosen as an interface which will be used to collect the information about different variables in the generic model. As shown in example of Figure 3.10 all choices from vehicles mass to size of tyres are made in Excel environment to ease up modeling process. Simulink and Python script will read cells from Excel and work according to the values given. In example of Figure 3.10, except dimension input cells, all other cells are drop down menus. From these drop down menus user can choose which product sets are included to the model.

Choose model properties

- Environment	Dome
- Body	Valtra
- Wheel type	Valtra_T202
Amount of wheels	
- At front	2
- At rear	2
- Motor	Drive
- Drive	4WD_1



Origo is at the centre of rear axle.

Give following values

Distance between axles (1)	2748	mm		
Distance between front tyres (2)	1830	mm		
Distance between rear tyres (3)	1810	mm		
Front tyres:				
- Diameter	1168	mm		
- Width	500	mm		
- Mass	195	kg		
- Inertia	23.2	0	0	
		23.2	0	
			44.7	

Figure 3.10 Example method of collecting infromation with Excel datasheet.

3.7.6 Simulink

Although Simulink will not be included to this master's theses it is still really important part of the generic model and its role will be shortly explained. Simulink is block diagram environment which is integrated to MATLAB and it can be used for model-based design and multidomain simulation. It is used in this project because it can be connected to hardware for real-time simulations. [24]

Simulink will be used to communicate with MeVEA simulation model and also to run hardware which is connected to simulation. As told before Simulink will read initial values from Excel file and then it will communicate and exchange information with MeVEA. Some components are simplified for MeVEA and Simulink makes accurate calculation for them. Simulink will then give calculated values back to MeVEA which will use them for simulation. Simulink also

runs hardware which is next to simulator platform according to simulation. All of this will be done in real-time.

Simulink is also used to simulate working of electrical components like battery packages, motors and generator. Also working of hydraulic systems will be simulated in Simulink environment.

4 CASE STUDY: AGRICULTURAL TRACTOR

With complete generic simulation model it is possible to build different simulation models with different tractor in seconds. Simulation models can vary from tractors with normal combustion engine to one which has electric driveline system with electric motors on each wheel and log trailer behind the vehicle like the one in Figure 4.1.

Figure 4.1 Results of generic simulation model. Tractor is modeled with Valtra's cabin, extra pair of wheels at the rear and with log trailer.

4.1 Simulated vehicle

Generic simulation model is mainly meant for marketing purposes for electric motors for vehicles. Electric driveline is good solution for vehicles which needs high speed and high torque during their work cycle and which can offer biggest market opportunities. First vehicle is wheel loader which needs huge torque when it is used to move loads around and high speed when it is moved between working locations. Second ones are tractors which are used worldwide for heavy labor. Third ones are harvesters which need high torque when moving in rough terrain.

Because of the large scale of the work and limited resources it was decided that all focus will be given to one type of vehicles, tractors. First generic simulation model will be made for tractors and it can be used to model different type and size of tractors really quickly. While one specific kind of vehicles will be used for making of generic simulation model it is important not to design generic model only for tractors. When simulation possibilities for different tractors are wide enough then generic model will be improved by adding other products to it. This is why generic simulation model needs to be easily modifiable for further usage.

4.2 Working principle of tractors

Tractors are commonly used heavy working vehicles and according to World Bank, World Development Indicator and EconStats there were over 28 000 000 agricultural tractors worldwide in 2006. [25] While tractors have generally same working principles as cars they are still different in many ways. Cars are mainly used for transportation and they only need to move themselves and a small load to destination as fast as possible. While cars are designed to move at high speed all capabilities of the engine cannot be used. This is because of restrictions and speed limits. Tractors do not have to move fast and they are mainly meant for heavy labor in which engine should produce high torque which will be used to move the vehicle. With proper reduction gear ratio it is possible to use all capabilities of the engine and driving in a rough terrain is possible. [26]

From engine power will be applied to the wheels trough transmission which includes clutch, gearbox and differentials. When starting tractor, engine needs to be under minimum load. After engine has started and is running it can work under bigger load and can be connected to transmission. Transmission and engine can be disconnected or connected to each other with a clutch. With a clutch it is possible to keep engines load minimum at start up or when changing gears. Gearbox is used to change gear-ratio that affects on the rotation speed of the wheels and amount of torque that will be transferred from the engine to the wheels. It is heavy and sturdy box full of gears which makes it possible to use variety of different gear-ratios. It is located close to the engine and is also often used as part of the frame. [26], [27]

After gearbox torque goes to differentials which divides it equally for both rear wheels. It can also be used as reduction gear. Differentials will divide torque so that each rear wheel gets equal amount of torque. When tractor is steered it also makes it possible for both wheels to rotate in different speeds. Because of this wheels will not wear so much because they will not oppose the movement of another. Differentials can also be locked to get both wheels rotate with same speed. This can be really useful on unpaved terrains when one wheel is on the slippery surface. [27]

Before wheels there is reduction gear which decreases the rotation speed of the shaft and increases the amount of torque which will be transferred to the wheels. Reduction gear is placed right before wheels to decrease the mass of the whole transmission system. If it would be located right after gearbox all shafts and other components would need to be bigger to withstand bigger torque. Most of the time reduction gear is made with planet gears. [27]

One possible way to build tractor is to install all components like wheels, cabin, engine and gearbox on the frame which is the foundation of the tractor. Frame will hold parts together and in correct relation to each other. There is also a possibility to build tractor without frame but in that case all parts and components needs to be designed so that they will fit together. This way components themselves will work as a frame. [26]

Hydraulics is an effective method to produce big forces and that is why it is commonly used in heavy labor vehicles like tractors. In tractors hydraulics is used to control hitch and many arbitrary and additional labour equipments like frontal or rear bucket. Hydraulic system includes for example hydraulic oil storage, pump, hydraulic hoses and cylinders.

4.3 Collecting model information

Collecting model information is done with Excel datasheet which is shown in appendixes of this master's thesis. Depending on what kind of answers are wanted information is collected either with dropdown menus or with empty cells which can be filled with numerical values. Dropdown menus are used to ask overall questions about model like what kind of environment or motors

will be used. Most important numerical variables are also asked. Those are dimensions, locations, masses and inertias.

Decision tree in Figure 4.2 shows what kind of things are asked with Excel datasheet. Simulation model will be build according to the decisions user will make and what kind of values he gives. In Figure 4.2 boxed out items are chosen from the dropdown menus while other information like inertias are needed as numerical values. Numbers which are shown in chart describes different possible solutions which can be chosen for that product. At the moment there is not many alternative solutions for some areas of simulation. For example at the moment there is only one set of wheels which can be chosen and only one body type which can be used. For some other areas of the simulation model there is more solutions to choose from. One example is motor model. If tractor is run by electric motors user can decide the amount of the motors which are used. Four-wheel-drive vehicles can be driven with one, two, three or four motors.

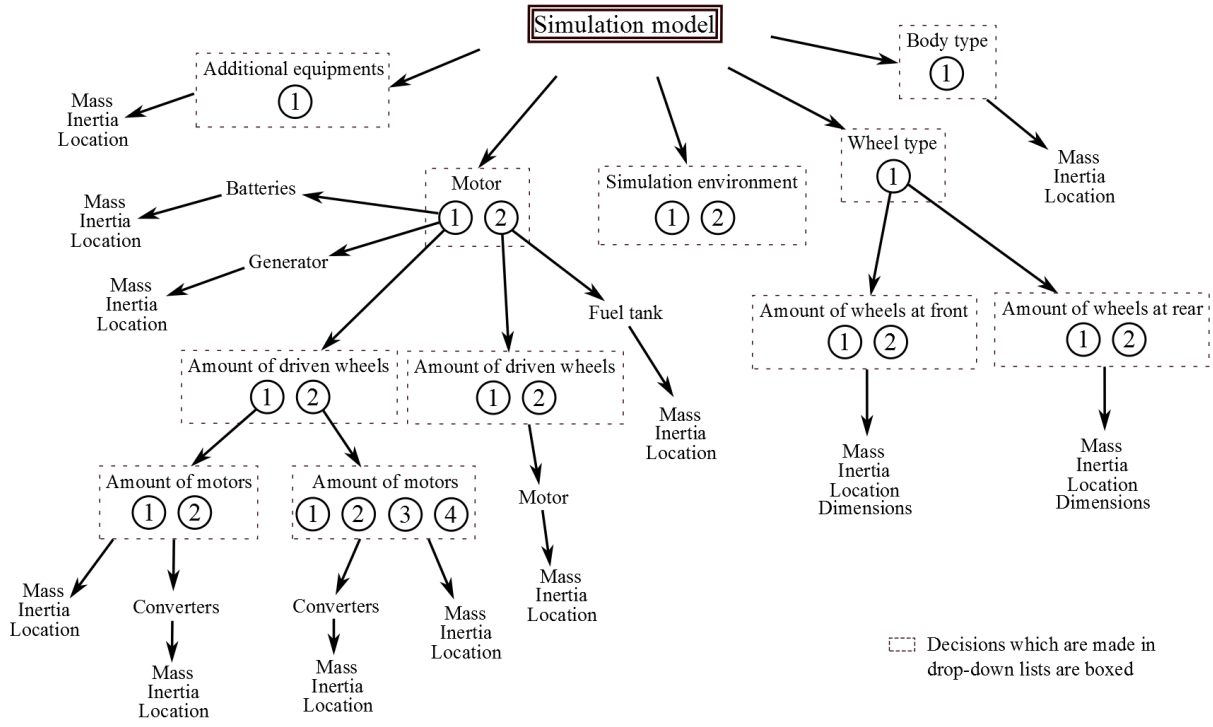


Figure 4.2 Chart about all the decisions which can be made in Excel. Boxed properties are chosen form the drop-down lists.

Figure 4.2 also shows how different decisions affects simulation model. For example there is two different type of motor solutions. Vehicle can either use combustion engine or electric motors to move forward. If user chooses combustion engine then he also has to decide mass, inertia and location of the fueltank and motor. Also the amount of driven wheels needs to be decided. If vehicle will be modeled with electric motors then user has to give mass value, inertia and location for the battery package, motors and converters.

All values which are collected with Excel are requested in specific units which are shown next to the answer box. For example dimensions are in millimeters and weights in kilograms. These units are chosen because they are widely used by different manufacturers.

4.4 Constructing simulation model from parts

MeVEA will save simulation models in three different files from which .mvs –file is main simulation file. That file will be used to load simulation model in MeVEA. From .mvs –file MeVEA will read names of two other files which will also be read. First file is .xml –file which includes all properties and data of the simulation model. This is the most important file in whole generic simulation model and it will be rewritten for every simulation. Second file which will be read is world –file. World –file includes all camera and light options of the simulation model as shown in Figure 4.3.

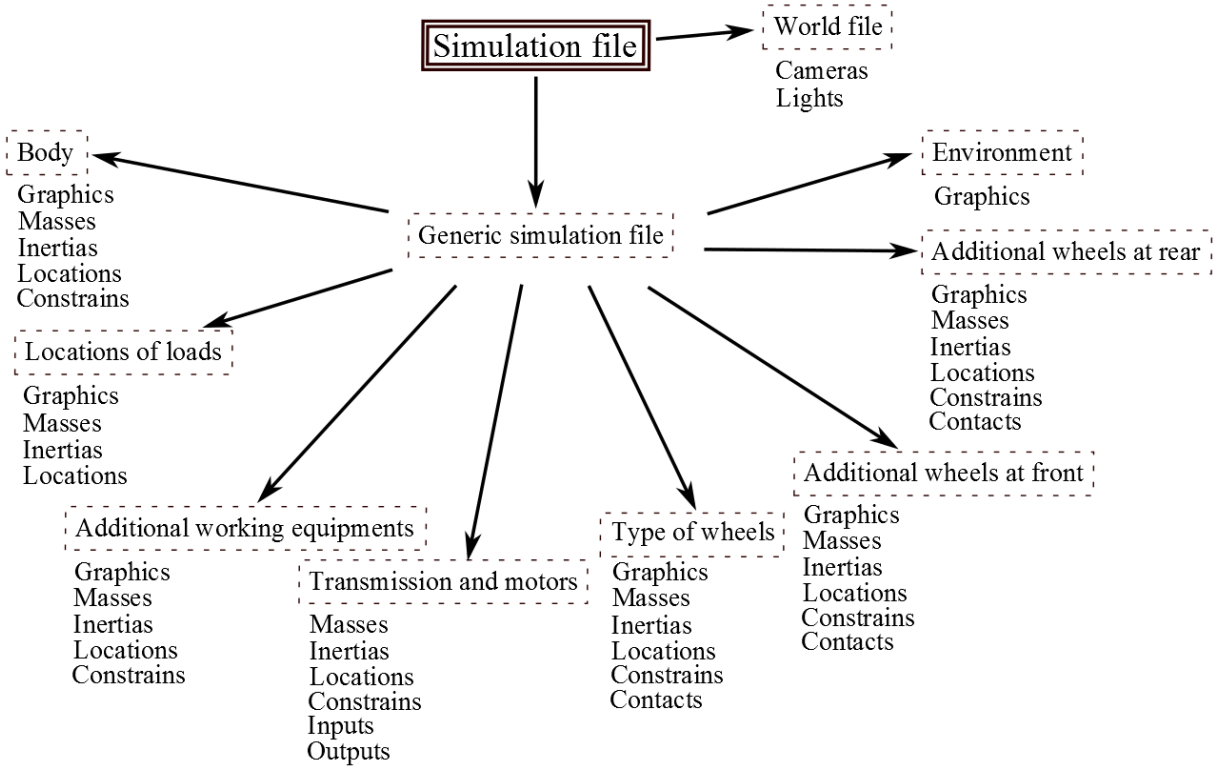


Figure 4.3 Generic simulation model is build from several smaller pieces which will change depending on choises user makes on the datasheet. Chart shows all eight filegroups which will be read by generic simulation file.

Figure 4.3 shows how generic model is divided to different files and how these files will read each other. Normally almost all information of the simulation model would be written in the generic simulation file which is shown in the middle of the chart of Figure 4.3. Because modifying big and complex text file is not easy generic simulation file is divided to eight different kind of file groups which can include several sub-assemblies. Generic simulation file is only used as collection platform for all sub-assemblies and it only includes command lines that one file from each file category will be read. There are eight command lines in generic file that defines which sub-assembly file should be read. For example there are two different sub-assemblies for environment. One includes all necessary information about gravel pit and other one includes all information about dome like environment. Depending on what kind of answer is given to Excel datasheet only one from the two environment options will be chosen.

There are two kinds of sub-assembly files. First ones are main files and they will be changed depending on the choices user makes in Excel. These files are over written every time new simulation model is made. They are also ones which will be used when simulating. Second kind of files are base files and as the name tells these are used as base when model is build and will not be changed during modeling process. Base files are only used to make new main files and they are not used when vehicle is simulated as shown in Figure 4.4. These two kinds of files will be explained further in chapter 4.5.

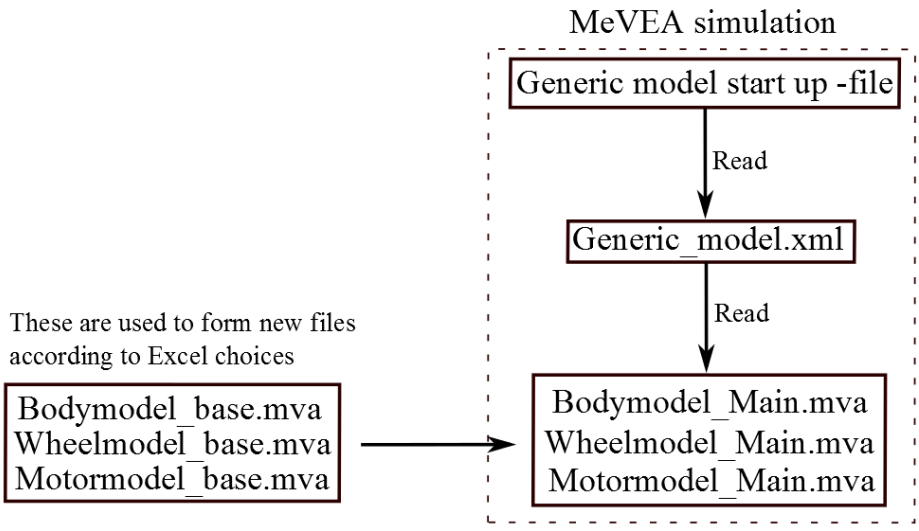


Figure 4.4 Base files are only templates which are used to form simulation files. Main files which are used in simulation are replaced with new ones every time new generic model is made.

4.4.1 Environment

There are two kind of environment models. First one is gravel pit and second one is simple flat terrain. Both environment types are modeled in their own sub-assembly files from which one will be used in every simulation. Flat terrain type is named as dome –model because of the dome like shape caused by the sky graphics. Dome environment has been modified from the environment of MeVEA tutorial 3. All fences, dumpster and other items has been removed so that only three different graphics remains. Those are ground, sky and tree line which surrounds the area. Gravel pit simulation environment is also from MeVEA Ltd. but it is much more

complex than dome model. For example it has movable rocks, shed, concrete barriers, trees, containers and other objects. Gravel pit environment has been modeled as one sub-assembly file which can be chosen from the Excel datasheet.

4.4.2 Body model and wheels

At this point there is only one choice for body model which includes both cabin and whole outer frame of the tractor as shown in Figure 4.5. Body model do not include mass of various main components like combustion engine and fueltank. Those are added in the simulation model from other sub-assembly files. Combined mass, inertia and location of cabin and frame can be modified from the Excel.

Figure 4.5 Generic simulation model has been divided to smaller sub-assemblies. One sub-assembly includes outer frame of the tractor and cabin.

There is also only one wheel set which can be chosen but it can be modified in more ways if compared to the body model. Assembly file for wheels includes both front and rear wheels, front axle, contacts between wheels and ground, right and left pivots with mudguards for front wheels and tie rod. Sub-assembly file for wheels is modified mostly when compared to the other sub-assembly files. From Excel datasheet it's possible to decide width, diameter, mass and inertia for rear and front wheels. Most important distances can also be changed. These are distances

between axles, distance between rear wheels and also between front wheels. Wheels and their contact surfaces are defined with splines. Sizes of the splines will also change depending on the choices which are made in Excel.

It is possible to add extra pair of wheels to rear or front axle. This is done with two more sub-assemblies, one for rear wheels and one for front wheels. According to Excel choices additional wheels are added to the tractors. These extra wheels are also updated according to the Excel.

4.4.3 Motors and transmission

At the moment there are two different possibilities for motors. The first one is to use combustion engine and the second one is to use one or more electric motors. For the transmission there are also several different possibilities depending on what kind and how many motors are used. For sub-assembly files motor type and transmission will go hand to hand and different sub-assembly files are combinations from both of these.

For combustion engine there is three different sub-assembly files to choose from. First option is two-wheel drive (2WD) in which combustion engine will produce torque to drive shaft through gearbox. From that torque will go through differentials to rear axle. From rear axle torque is transferred through planet gears to the wheels and vehicle will start to move. Second sub-assembly file is also 2WD and includes same components as previous one but this time torque is transferred to front wheels. Last option when combustion engine is used is to use four-wheel drive (4WD). In 4WD torque is used to drive all four wheels of the vehicle.

There will be several different option more if electric motors are used. Most interesting would be 4WD in which motors are installed in every wheel of the vehicle as shown in the example wheel loader of Figure 4.6. In that case torques for every motor are calculated in Simulink and then given back to simulation model in MeVEA. Calculated torques are then transferred through planet gears to the wheels.

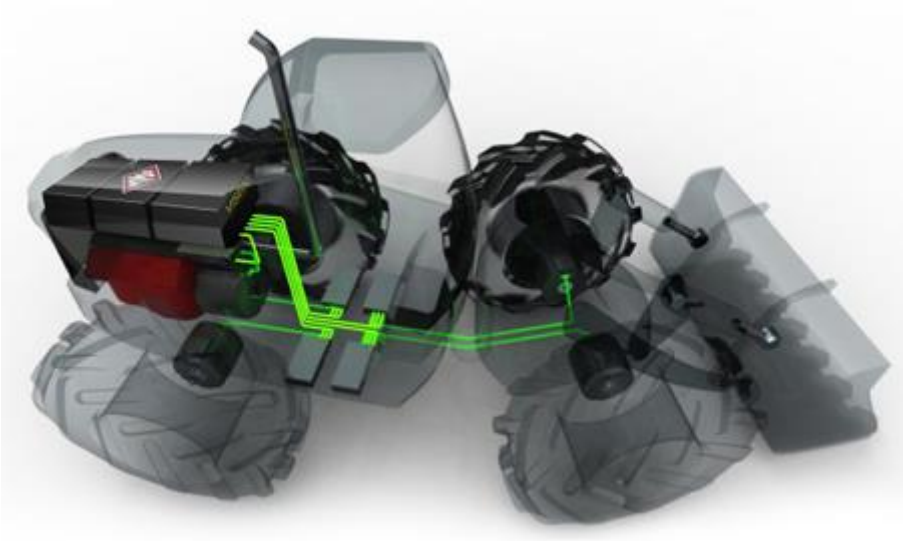


Figure 4.6 Electric drivetrain of wheel loader which is run by several electric motors. [28]

User can also choose other kinds of 4WD and 2WD electric driveline systems from the Excel datasheet. They can include one, two, three or four electric motors. Following lists shows all different possibilites which can be chosen from Excel datasheet as motor and transmission system. All different options are modeled in their own sub-assembly files.

With combustion engine:

- 2WD – Engine is used to drive rear wheels
- 2WD – Engine is used to drive front wheels
- 4WD – Engine is used to drive all wheels

With electric motors:

- 2WD – One motor is installed on the rear axle and it will drive both rear wheels
- 2WD – One motor is installed on the front axle and it will drive both front wheels
- 2WD – One motor is used to drive each rear wheel
- 2WD – One motor is used to drive each front wheel
- 4WD – One motor is used to drive all wheels
- 4WD – One motor is used to drive rear wheels and other one to drive front wheels
- 4WD – One motor is used to drive front wheels. There is also motor on each rear wheel

- 4WD – One motor is used to drive rear wheels. There is also motor on each front wheel
- 4WD – There is one motor on each wheel of the vehicle

4.4.4 Accessories

Extra working equipments are divided intentionally as a group of their own. This is to make adding of new products easier. At the moment there is only one additional working equipment which can be used in simulations. That is log trailer which will be connected behind the tractor. Log trailer is modeled in one sub-assembly file which includes all necessary modeling information about the product. Those are for example information about graphic files, properties of trailers wheels, wheels splines and collision information, masses, inertias and connections between different parts. Log load of the trailer is described with a dummy of which location and mass can be changed from Excel datasheet.

4.4.5 Simulating objects with dummies

One way to define parts in MeVEA software is to use dummies. Dummies are parts which are relatively light to be simulated and it is recommended to use them when it is possible. Dummies have their own mass and inertia values. They are not connected to the kinematic chain but they will still affect mass and inertia properties of the attachment body. [29]

Most dummies are collected in one sub-assembly file which can be easily modified. Inertia, mass and location of every dummy can be changed from Excel datasheet. All value changes in Excel will automatically update sub-assembly file and dummies properties.

At the moment there are 14 dummies in one sub-assembly file. Those are fuel tank, front and rear loads, battery package, generator, four motors and five converters. Each of these are highly modifiable.

4.4.6 Graphics

In MeVEA graphics are only used to make parts visual and their only job is to be seen. Graphics do not give any properties for the parts and they only visually demonstrate how objects will move. Because of that there is possibility that graphics can sometimes overlap.

Since MeVEA is not designed to make graphics it do not have many graphic modification capabilities in it. Graphics are made with other software applications and saved in specific file format. MeVEA will then read those files and use their graphical data in simulations. MeVEA only has information about which graphic file is used and what is its location and orientation in simulation. Graphics are connected to some object and they will move together.

Tractor's graphics that are used in generic model are received from MeVEA Ltd. with permission from tractor manufacturer Valtra. Generic model uses graphics of Valtra T202 tractor in simulations. This includes tractor's cabin, outer frame, front and rear tyres and rims, tie rod, front axle and mudguards.

Environment graphics are also from MeVEA Ltd. As told in chapter 4.4.1 dome –environment with flat terrain is modified from MeVEA tutorial 3 by removing some objects from the model. At the moment only graphics for ground, tree line and sky are used. More complex environment model of gravel pit and all its graphics are also received from MeVEA Ltd.

4.5 Generic model assembly tool

Generic simulation model is made as automatic as possible with a script. Script is made in python programming language and it builds simulation model according to the choices user makes in Excel file. Full Python script can be found from appendixes.

Python script is the most important part of the generic simulation model because it makes modeling process more automatic. The main idea of the script is that it will first read the Excel datasheet and then write new MeVEA simulation files as shown in Figure 4.7.

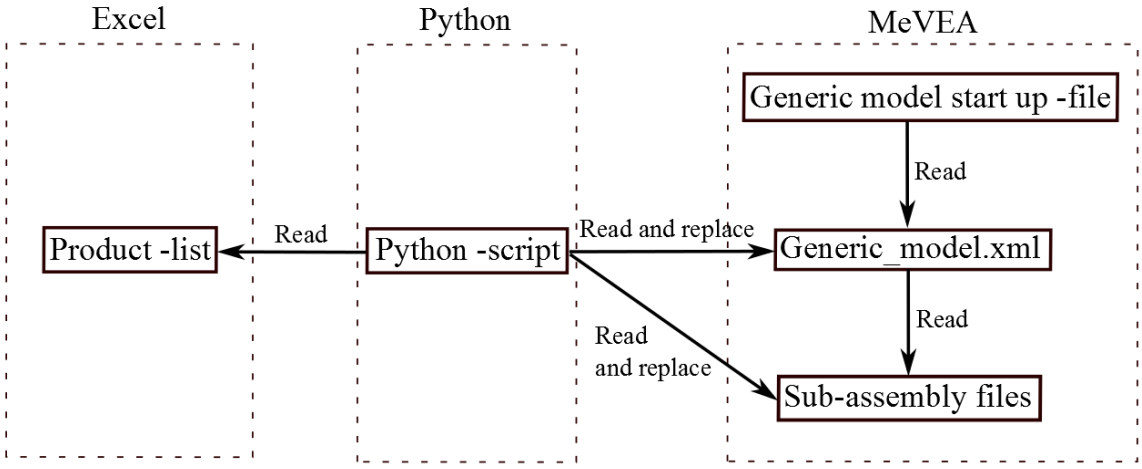


Figure 4.7 Communication chart between programs.

Basic working principle and structure of the Python script is shown in Figure 4.8. When script is run it will first download all necessary modules which are needed later on script to perform various actions. For example one makes it possible to read Excel file. Next a couple of functions will be formed to decrease the amount of coding which is needed later. These two things will be done every time new simulation model is made.

On the next step Python will read name of the file which needs to be modified from the Excel datasheet. It reads all text from that file and starts to replace old values with new ones. These steps are repeated until all necessary values from that file are replaced with new ones. File with new values in it will be saved in another name. This new file is the one which will be used during simulation. These previous steps will be repeated until all necessary files are modified according to the Excel choices as shown in Figure 4.8.

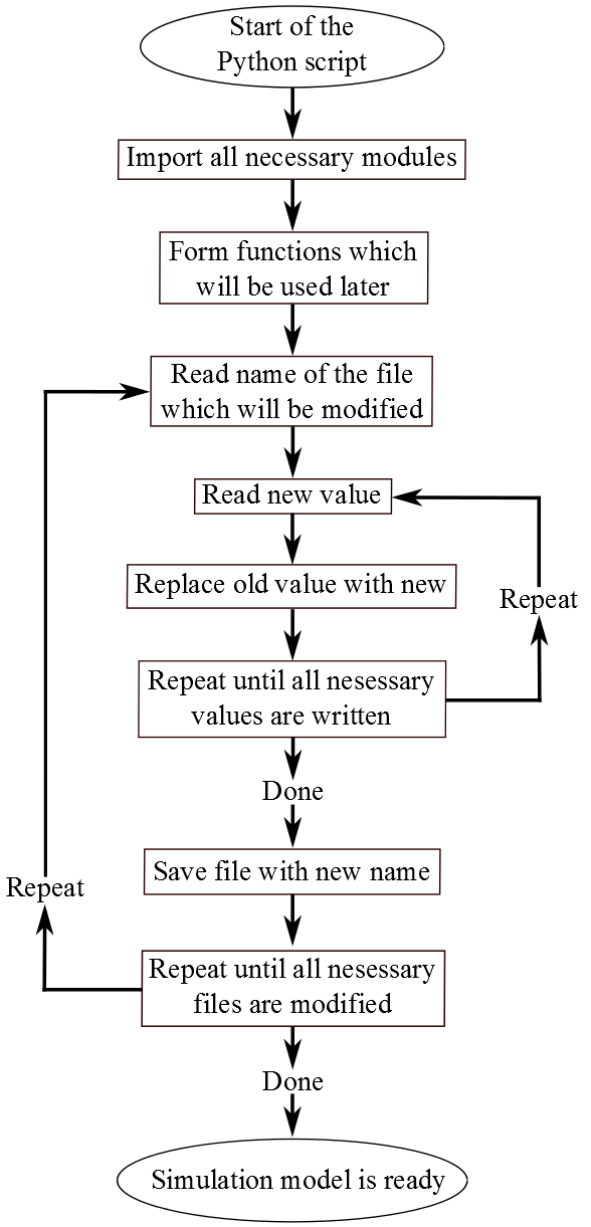


Figure 4.8 Simplified working principle of generic model script.

Now some small parts from the script will be inspected closer. After all necessary modules are loaded a couple of functions will be defined from which one is shown on Figure 4.9. After defining the example function “make_assembly” it can be later called and used with just three lines of script. This will greatly decrease the amount of coding. All defined functions has almost

same kind of purpose but they will do it in slightly different way. Their job is to replace specific line of marks with another line of marks.

```
21 def make_assembly(BodyType, searcheable, replacement):
22
23     mva_handle = open('.\\Assemblies\\' + BodyType.replace('_Main','_base'),\
24     'r')
25     mva_content = mva_handle.readlines()
26     mva_handle.close()
27
28     mva_contentOld = mva_content
29     mva_content = []
30
31     for line in mva_contentOld:
32         line = line.replace(searcheable, replacement)
33         mva_content.append(line)
34
35     filu_mva = open('.\\Assemblies\\' + BodyType, 'w')
36     filu_mva.writelines(mva_content)
37     filu_mva.close()
```

Figure 4.9 One of the functions used in generic model which is used to replace specific content with another content and to save file in another name.

Figure 4.9 shows how one of the functions which is used on generic model is defined. Its job is to make a copy from one of the files in folder "Assemblies". Script will remove "_base" –text from the name of the read file and replace it with "_Main". Before saving the file with a new name function will search specific line of marks and replace it with another line of marks. Figure 4.10 shows one example of how function of Figure 4.9 is used later on the Python script.

```
83 WheelMass = str(float(sheet.cell(25,3).value) + 0.001)
84 BodyType = 'TyreSet_' + str(sheet.cell(7,2).value) + '_Main.mva'
85 make_assembly(BodyType, 'Replace_TyreF_mass', WheelMass)
```

Figure 4.10 Mass of the front wheel will be written in simulation file with the help of previously defined function of Figure 4.9.

Example of Figure 4.10 shows how mass of the front wheel will be written in simulation files. Script will first read how heavy the wheels should be from the specific cell of the Excel file and then 0.001 kg safety value will be added to it. This safety value is used because every body in MeVEA needs to have a mass. If user do not give a mass for object in Excel datasheet it is still

possible to run simulations because script add a minor mass for each part. The mass that is read from the cell (25,3) will be then saved as variable “WheelMass”. Next script line will read another cell from the Excel from where it gets the name of the file which will be modified. The last line of this script uses previously defined “make_assembly” –function to form new TyreSet_Valtra_T202_Main.mva –file from the old TyreSet_Valtra_T202_base.mva –file. Before saving file with the ending “_Main.mva” script will change content of the file by replacing variable “Replace_TyreF_mass” with mass of the wheel.

There are a total of four different functions defined at the start of the Python script. Everyone works in a similar way as the one told before but with minor changes. For example there are several different variables in TyreSet_Valtra_T202_base.mva –file which need to be changed to correct values. Because previously explained function will always replace only one variable and save the file on top of the old _Main.mva –file it cannot be used for all variables. After “make_assembly” –function has been used further changes will be made with simpler function which will not affect on the previously made changes.

All sub-assembly files which are named with the ending “_base.mva” contains variables which will help to identify where different values needs to be placed. “_base.mva” will never be changed because only a copy from it will be modified, saved and used.

Main generic model –file is slightly different than sub-assembly files. In sub-assembly files variables include for example masses, locations and inertias while in main generic model file script will change only file paths as shown in Figure 4.11. Generic model file has command lines written in itself which tells it to read specific files in specific folder. Script will change these file paths so that correct files will be read. For example if simulation environment is gravel pit then script will modify main simulation file so that it will read correct sub-assembly file from correct folder.

```
<Assembly FileName="Assemblies/Environment_Gravel_Pit_Main.mva" PreFix="">
  <!---->
</Assembly>
<Assembly FileName="Assemblies/Cabin_Valtra_Main.mva" PreFix="">
  <!---->
</Assembly>
<Assembly FileName="Assemblies/AdditionalEquipment_Log_Trailer1_Main.mva" PreFix="">
  <!---->
</Assembly>
<Assembly FileName="Assemblies/TyreSet_Valtra_T202_Main.mva" PreFix="">
  <!---->
</Assembly>
```

Figure 4.11 Generic model file includes command lines which will tell which sub-assembly files needs to be read.

Figure 4.11 shows an example of how and what kind of sub-assembly files can be used in simulation. These command lines are written in Generic_model_Main.xml –file and they order to read sub-assembly files like the one which includes gravel pit environment model. Command line tells that this specific environment model can be found from Environment_Gravel_Pit_Main.mva –file in Assemblies folder. Example code of Figure 4.11 also loads Valtra's cabin model, adds log trailer behind the tractor and installs specific tyres to the tractor.

4.6 Simulink connection

Simulink will only be dealt briefly because making of the Simulink environment for generic model is not part of this master's thesis.

Simulink is only used to simulate working of electrical components during simulation. These can be for example charge level of batteries, torque caused by the electric motor or yield of electricity of generator. When electric motors are chosen from Excel datasheet all of previously mentioned products are simulated. In one case there can be four electric motors which will be driven according to the driver. Each of these motors are simulated individually.

At the moment MeVEA and Simulink can communicate with each other with inputs and outputs via socket-interface. MeVEA model will give Simulink model realtime data about position of the throttle and clutch which will be used together with the information it gets from the Excel

datasheet. Simulink will simulate behavior of all electrical components with the data it gets from those sources and give it back to MeVEA. MeVEA model has a couple of inputs which will be used for Simulink model to give information about torques given by each motor. Depending on the simulated model MeVEA will transfer these torques to the wheels through differentials and planet gears.

4.7 Validation of the generic simulation model

Working of generic simulation model and its connection with Simulink model was tested with a quick test. Generic simulation model was used to build two tractor models which had electric transmission with electric motors on each wheel. This was also to test working of the Simulink model which was made to simulate working of the electric components. Both vehicles were modeled exactly the same but the other one was pulling trailer of the weight of 12 000 kg as shown in Figure 4.1.

After driving around the simulation environment with both vehicles Simulink gave simulation results like the one in Figure 4.12. Figure 4.12 shows one example of what kind of data can be collected during simulation. This specific one shows efficiency of the motors during the simulation. Results of the test simulation shows that generic simulation model works well and it can be used to simulate working of different vehicles together with Simulink. Most astounding is the speed in which it can make completely new models. After Excel datasheet has been filled and user has made all decision about the simulation model a new model will be built in a couple of seconds.

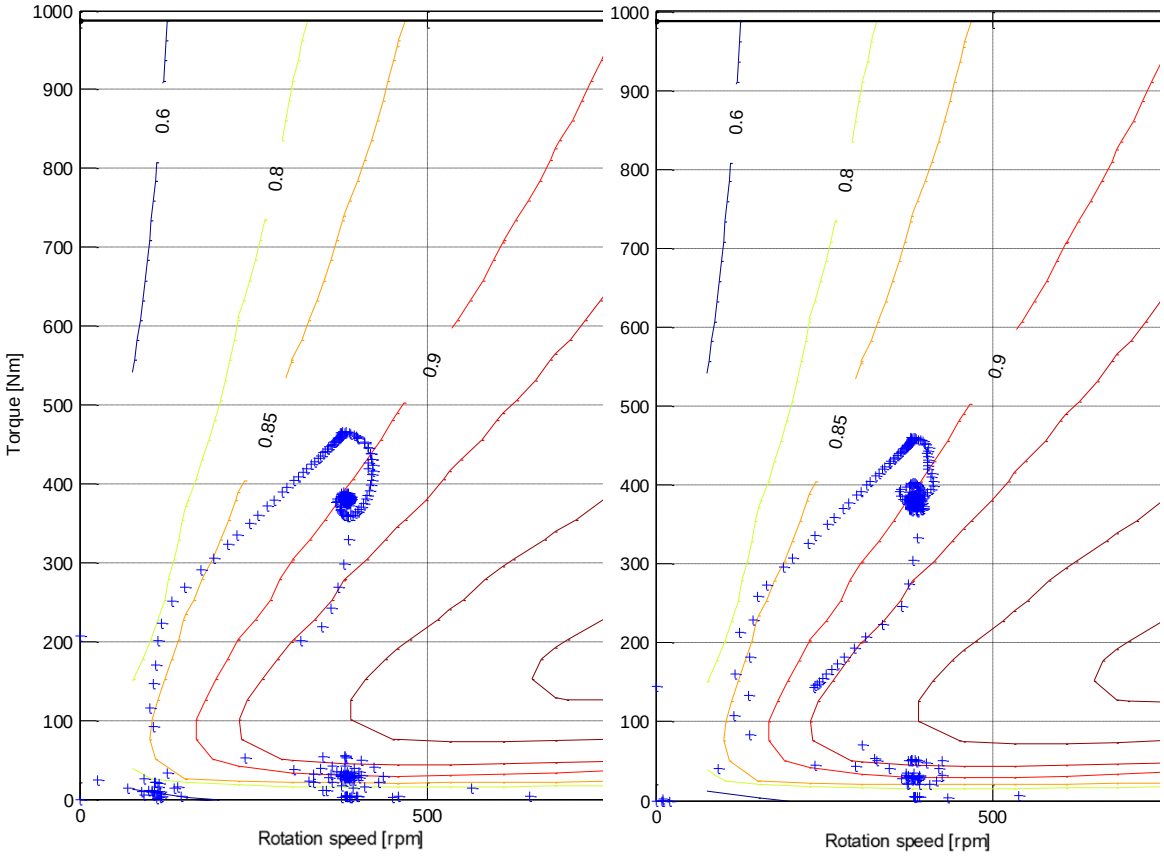


Figure 4.12 Efficiency maps from two different simulations. Electric motors was used in both simulations and in each wheel. In second simulation tractor also pulled 12 00kg trailer.

5 ANALYSIS

Generic simulation model has achieved most of expectations which were given at the start of the project. It has also over exceeded some of those expectations like the amount of time which is needed to build new simulation model with it. Developed generic simulation model is really capable and it can be used to produce various accurate simulation models easily and in short notice. One noteworthy downside of the simulation model is graphics which are not generic at the moment. Making of new graphics and in correct file format will take time.

5.1 Range of the different simulation possibilities with generic model

Generic simulation model can be used to simulate variety of different tractors in different environments. At the moment there are two different environment models to choose from and each of these can be used to test vehicles in variety of ways. They offer big and flat areas and also hills and slopes.

Tractor model itself can be easily changed to match different types. If it is necessary whole model can be changed significantly. For example there is no limit for the size of the wheels and their diameter, width, mass and inertia values can be freely changed. Also location of every wheels can be changed freely. Same things can be also done for cabin, frame structure and every other part of the vehicle.

Simulation model will be mainly used to test and show differences between tractors which are with combustion engine and those which are run by electric motors. It is also possible to model vehicle which uses combination of those. In this area generic simulation model is working well because vehicles simulation model can be easily changed between these two model types. Biggest changes can be found between models which uses combustion engine and ones which uses electric motors.

Because most of the modelled parts are movable it is possible to use simulation model to research how different parts should be placed in the vehicle. Simulation model can be used to find out for example how electric motors, converters and battery packages locations and sizes will affect behavior of whole vehicle.

At the moment there is one additional working equipment modeled which can be added to the simulation model behind the tractor. That is log trailer. Because log trailers properties can also be changed it is good way to test how tractor will behave in different situation while pulling heavy loads. Braking and accelerating on the slope when log trailer is attached on the vehicle can offer valuable information. This information will be used for example to calculate how big electric motors and energy storages are needed on specific vehicles.

Connection between MeVEA simulation model and Simulink offers more possible variations for vehicles and their simulations. Now most of the electric components are simulated in Simulink environment because it can offer more detailed simulations for them than MeVEA. Thanks to the Simulink it is possible to run each wheel of the vehicle with different torque depending on the situation. If some wheels are slipping then power can be redirected to those wheels which have good grip and are not slipping. This is just one example of how Simulink connection can be used as an advantage.

While generic simulation model can be used to make many kind of different simulation models there is one minor flaw which do not have an effect on actual results of the simulation but it has effect on the visual outlook of the whole simulation. That is graphics. At the moment there is only one set of graphics for tractor which is from Valtra T202. Graphics cannot be made with MeVEA and it uses them by connecting graphics to specific parts in specific location. Including those graphics can only be scaled up or down and that is all. That will make simulation model look faulty in some cases. One example can be wheels. If user increases distance between rear wheels in some point it can seem like wheels are not connected to the tractor anymore. Tractor and wheels are working correctly during simulation but visually wheels are not connected to the vehicle. Model which looks unrealistic can have negative effect on customer.

All in all generic simulation model is highly modifiable and can be used for many different kind of simulation. However if generic model cannot answer to some simulation needs its database can be easily increased. By adding one more sub-assembly for example for wheels modeler will increase the amount of different possible simulation models substantially.

5.2 User interface and –environment

In users perspective using of simulation model is made really simple. At first user will start by opening Excel file where default choices and values are already given. User can change values from Excel as he wants or go by default values. Each of the cells in which values will be inserted has clear notes of what kind of values are needed. Some choices are made from clearly marked dropdown menus.

After Excel file is filled it will be saved and python script needs to be run. A script will instantly and automatically make new simulation files for MeVEA according to the choices user made to the Excel file. Only thing that remains for user to do is to open simulation files with MeVEA software and run simulation.

User environment for further modelers is not so simple. When new possible choices are included to simulation model it is important to take notice of the old ones too because python script is using replace method. This means that script reads file and looks for previously defined line of markings which can include numbers, letters, slashes, dots and other markings and will replace them with previously defined line of markings. This way it is possible to change files which will be included to MeVEA simulation file. Because this replacement method script will change all marking lines which fit to the description it is important to pay attention to the naming of the replacement lines. Variables needs to be named so that they cannot be mixed up to other text.

Because generic model is splint to different sub-assemblies modeler can focus on small areas and improvements at a time. Clearly divided sub-areas will help to understand the behavior of the simulation model and different components. For example if modeler wants to increase the

amount of different wheels there is no need to reform whole model but to make new sub-area file under wheel -category. Dividing model to several small sub-areas also helps with troubleshooting.

For graphics there is not really any shortcuts. New graphics cannot be made automatically with scripts and old ones can only be scaled up or down in MeVEA. New graphics needs to be made from the scratch and saved in .3ds format for MeVEA. It is a good thing that there is rarely a need for new graphics because when vehicles with new and old driveline are compared graphics will be same or change only a little.

Overall generic simulation model is user friendly. New simulation models can be built by using simple Excel file which has every modifiable values marked clearly in it. Also the possibility of using Excel wrong is noted. The risk that user will give faulty values to the Excel file is been decreased. Generic simulation model is as user friendly as at the start was hoped for.

5.3 Suitability for marketing

Including research generic simulation model can also be used for marketing purposes. One of its advantages is astounding speed to generate new simulation models. This is really good when results needs to be gotten fast like for example in conferences or when customer will visit in short notice.

Thanks for all modifying possibilities of the generic model it is possible to build simulation model from customers own vehicle. Customer can use simulator and drive their own vehicle to test simulation environment and all the capabilities of the simulation software and model. This way he can verify accuracy of the simulation model which will make advertising of the product easier and further results will seem more trustworthy.

When customer has confirmed accuracy of the simulation model a real marketing phase can start. At this phase advertised product will be included to the simulation model which is made from customer’s vehicle and simulations will be driven again. Driver of the simulated vehicle

can feel firsthand how behavior of the vehicle changes when new modules are installed on the vehicle. In this case there is also physical component next to simulator and it will move according to the simulation.

After two simulations with two different machines it is possible to compare simulation data between different vehicles. Customer can compare products for example in fuel consumption, noise level and in many other areas. Maybe one of the most important information which can be calculated from the results is payback time. While customer is told about all the advantages of the new components he also gets to know how long it will take for investments to pay themselves back.

6 FUTURE WORK

There is some work which needs to be done so that simulation model seems more reliable for the customer. Some graphics needs improvement and modifications and also some additional working equipments needs to be modeled to make testing of the model in different situations possible. Also some software changes would make further modeling easier.

6.1 Software applications

Most of the used programs work well together and they do not need big changes. There is one area which needs improvement and that is graphics and all software applications which are related to making of those. At the moment graphics are made with SolidWorks which will also give inertia values of every part for MeVEA. Because SolidWorks cannot save files in .3ds file format for MeVEA to use they need to be modified with third software which is called Blender. Blender is used to open SolidWorks files and save them in another format which can be read by MeVEA.

Using of SolidWorks and Blender combination to produce graphics has many downsides. First of all it would be best if graphics could be done with one software only. That would make whole modeling process faster and easier. Another downside of those two programs is about how Blender will read SolidWorks files. Blender do not recognize any colors which are used in SolidWorks and so that when graphic files are imported to Blender they are all gray. All coloring of the parts needs to be done with Blender. Also modeling parts with sheet metals in SolidWorks will cause some problems in MeVEA. MeVEA do not recognize correctly graphics which are made with thin layers and that can be seen as partially transparent graphics.

It is advisable to reconsider previously mentioned two software applications. If it is possible they should be replaced with one software which can be used to make graphics files and to save them in correct file format.

6.2 User-friendly choices

Making of changes to simulation model is made easy with Excel datasheet. It is quite clear and easy to understand already but it can still be improved a little by changing its layout. However it is recommended to improve generic model in other areas because results from changing of the layout of Excel datasheet is not worth the amount of time it needs.

Generic simulation model is quite simple for customers to use but at the start it can seem complex for future modelers who will increase the database of the model and widen different simulation possibilities. Despite the structure of the generic simulation model may seem complex it is made as simple and logical as possible. If simulation model will be modified further all new files should be archived and named in specific way to keep basic structure of the model tidy. This will also help other people to understand basic behavior of the model and to modify it easily. For example if someone wants to make new transmission or body solution for the model then sub-assembly files for those solutions needs to be named with the starting text of "Transmission_" and "Cabin_" because every other solution for same category is also named same way. This will help other people to find correct sub-assembly files.

Including naming new files correctly also adding of explanations and guides is important. It will not take much time to add simple explanation lines in the middle of python code or in the assembly files. Explanation lines which can help other users to understand what specific parts of the script does and that way decrease the amount of time they need for modeling process.

6.3 Visualization of the model

Graphics of the whole generic simulation model including environment are average. There is some visually accurate graphics in use but also some really poor ones. Valtra T202 tractor model is really accurate and its level of details do not need improvements. Actually it would be best to make tractors body and wheel graphics more simple to increase the performance of the simulation. Those graphics include details which could be simplified or removed without having a major impact on the outlook.

If tractors graphics are considered as highly detailed then trailers graphics are only average or poor. Coloring of the parts is done poorly with blender and they should be made again and better. Also logs could be made better to improve experience of which customer gets when using simulator. At the moment they look a little plain.

Simulation environment graphics also needs some modifications. At the moment dome –like simulation environment is really rough which will not have good effect on the overall impression that generic model will give for the customer. Dome model is good for testing purposes because its graphics are light but there should be alternative and more detailed environment for customers. There is already gravel pit environment model which can be used but flat environment is also needed.

Fine tuning of the model graphics can sound like wasting a time but it has important role in whole model. Customers rarely do know what kind simulation programs is used and what they are capable of. In this case they only see graphics and do not know what is beneath them. If graphics are poor they can falsely think that whole simulation is made poorly. Good graphics can have positive effect. There is also a risk that simulation model will give customer a game –like impression. This will lead to doubts and it can be harder to market products to them. These are main concerns which needs attention when old graphics are modified or new ones are made.

6.4 Further modification possibilities of simulation model.

Generic simulation model is built as modular as possible to make further modeling as easy as possible. It also helps in troubleshooting because in that case there is no need to check whole model for faults. Now that basic working principle of generic simulation model has been founded functioning it can also be easily used for other vehicles. These can be for example wheel loaders and harvesters which can benefit greatly from the electric motor which will be marketed with the simulation model.

For tractor model it is wise to build some other working equipments. Log trailer is working well but it would be good to have possibility to simulate other kind of working equipment also. Those

can be for example plough or frontal bucket or forks. Plough will inflict constant force to the vehicle and it will oppose movement. Frontal bucket and forks can be used for loading work. Modeling frontal bucket and forks for tractor will also make it easier to model wheels loaders later on. This is because it is easy to build wheel loaders bucket when there is working bucket model tractor which can be used as an example.

7 SUMMARY AND CONCLUSION

This master's thesis has been done for Drive! –project in which new and compact electric motor solution has been found. The main objective in this thesis was to build generic simulation model which could make real-time simulation models of mobile working machines for MeVEA software. These simulation models could then be used to research electric motor and vehicles even further and also to market new electric motors for customers.

It was decided at the beginning that generic simulation model needs to be able to produce new simulation models for different vehicles in short notice. Modeling speed needed to be high so that it is possible to build simulation models even from customers own vehicle as fast as possible. In this way customers can straightly see how they will benefit from marketed product. It was also decided that generic model would be made only for tractors but it should be also possible to modify it to include other working vehicles too.

MeVEA and Simulink software applications limited different possibilities of how generic model can be build. These two programs were chosen beforehand and they will communicate with each other in real-time during simulation. All electric components are simulated in Simulink and everything else in MeVEA. Generic simulation model needed to be designed so that it can produce simulation models which can be used by these two softwares. Also if there is some other software applications which are needed in generic model they also needs to work well together with MeVEA and Simulink.

Development process started by studying simulation softwares MeVEA and Simulink and what kind of simulations are made with them. People who had already used previous software applications were also interviewed and literature was studied. Research showed that it would be best to build simulation models for MeVEA without using simulation software itself during modeling process. This was made possible by the method in which MeVEA saves its simulation files, in file format which can be read and changed easily. It was also natural choice that

simulation model would not be built from the scratch but some kind of item library could be used to hasten modeling process. Finished generic simulation model uses small sub-assemblies to build full simulation model. It will pick correct sub-assembly files and also modifies them according to the user's decisions. For example masses, locations and inertias can be changed for most parts.

Modeling process starts by collecting information about buildable model. This is done with simple datasheet which can be given to the customer beforehand or it can be filled on the spot. Datasheet includes all changeable properties of the simulatable vehicle like motor and wheel type, cabin, transmission and many masses, inertias and dimensions. After datasheet has been filled a script will build new simulation model according to the answers given in to the datasheet.

The script will build new simulation model by using database which includes partially modeled sub-assembly files. Script will read correct files from the database and finishes them according to the user's decisions. Modified files will be then saved with another filename. Database of the generic model will not be affected and will stay same during and after the modeling process. The finalized simulation model will be overwritten every time new simulation model is made.

After datasheet has been filled and script has built simulation model it is ready to be used. During modeling process most of time will be spend to fill the datasheet while script will build simulation model in seconds. Extremely fast modeling speed makes it possible to use simulations to market products for example in product exhibitions in which each possible customer comes and goes as they please. There it is impossible to prepare to each individual customer beforehand and this method to generate new simulation models can be huge advantage.

As stated before generic simulation model can produce new simulation models from tractors in astounding speed. Generic model is comprehensive when it comes to changing of the tractors properties although there could be some more options for tractors graphics. At the moment new graphics cannot be made with generic model and vehicle can only be simulated with Valtra

T202 tractor's graphics. Also working cycle should be described better. There is possibility to add log trailer behind the tractor and to define its mass but vehicle should also be tested in different kind of work situations.

Generic model can produce really accurate simulation models when compared to modeling speed. Almost all properties of tractor can be changed easily from the datasheet. Simulation results can be easily plotted and shown to customer for example in a form of payback time, power efficiency of the electric motors and fuel consumption. Comparing different simulation results is easy.

Generic simulation model has fulfilled all initial demand which were given at the beginning of this master's thesis. It can produce quite accurate simulation models for tractors in short amount of time. This generic modeling process can also be easily used for other mobile working machines. Vehicles can either be simulated with normal combustion engine or with new electric motors. Mass and other properties of different parts like fuel tank, battery package and motors can be easily changed. Overall generic simulation model is excellent tool to market different product solutions for mobile working machines.

REFERENCES

[1] Chih-Ming, C.; Jheng-Cin, S., 2010, World Electric Vehicle Journal Vol. 4: Performance Analysis of EV Powertrain system with/without transmission, pages 629-634, downloadable from http://www.evs24.org/wevajournal/php/download.php?f=vol4/WEVA4-4090.pdf

[2] Ren, Q.; Crolla, A.; Morris, A., 2009, Effect of Transmission Design on Electric Vehicle (EV) Performance, downloadable from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5289707&tag=1

[3] Plug-in hybrid electric vehicle picture, Go electric drive –website, last modified on 13.11.2014, used 13.11.2014, downloadable from http://www.goelectricdrive.org/images/webParts/plugin_hybrid.gif

[4] Electric motor –picture, Visedo –website, last modified 18.10.2013, used 25.11.2014, downloadable from http://www.visedo.com/images/stories/products/permanent-magnet-motors/3-powerdrum-xxs-visedo_excavator_swing_drive.jpg

[5] Law, A. M.; Kelton W. D., 2000, Simulation modeling and analysis, 3rd edition, McGraw-Hill higher education, 760 p.

[6] Carson, J. S.; Nelson, B. L., 1996, Discrete-event system simulation, 2nd edition, Prentice-Hall, 548 p.

[7] Shabana, A. A., 1998, Dynamics of Multibody Systems, 2nd edition, Wiley & Sons 372 p.

[8] Amirouche, F., 2006, Fundamentals of Multibody Dynamics, Birkhäuse, 684 p.

[9] MeVEA Reference Manual, version 7.67 v15, 95 p., last modified 21.01.2014.

[10] MeVEA Ltd. –website, www.mevea.com, internet source, last modified 13.11.2014, used 13.11.2014.

[11] Loh, E.; 2003, Technical document: Hardware In the Loop Simulation Chassis System Applications, Ford Motor Company, 20 p, downloadable from http://www.opal-rt.com/sites/default/files/technical_papers/ford_sept2003.pdf.

[12] Simulator –picture, MeVEA Ltd. –website, last modified 5.9.2014, used 25.11.2014, downloadable from http://www.mevea.com/sites/default/files/ACS2DOF_2014_1.jpg.

[13] Valtra T-Series 155-250 hv -brochure, last modified 24.11.2014, used 24.11.2014, downloadable from http://www.valtra.fi/downloads/Valtra_T4_FI_print proof.pdf.

[14] Volvo picture gallery, http://images.volvoce.com/#1416828059760_4, last modified 24.11.2014, used 24.11.2014.

[15] Nokian Leader Grip 3- tyre picture, Nokian Heavy Tyres –website, last modified 13.6.2012, used 24.11.2014, downloadable from http://www.nokianheavytyres.com/files/nht/product_pictures/nokian-loader-grip3-iso.jpg.

[16] Kittilä mine –picture, Agnico Eagle –website, last modified 7.1.2014, used 24.11.2014, downloadable from http://www.agnicoeagle.fi/PublishingImages/Photos/ilmakuva%20avolouhoksesta%20ja%20rakennukset%20taustalla-E.jpg

[17] Electro Hydraulic control solutions for Wheel Loaders –brochure, last modified 20.11.2014, used 24.11.2014, downloadable from http://www.hydraforce.com/Literatu

re/Solution_Brochures/English/WheelLoader_Solution.pdf

[18] MeVEA Tutorial 2, 15 p., last modified 10.01.2011.

[19] MeVEA Tutorial 3, 14 p., last modified 01.10.2011.

[20] Tractor –picture, TUMOSAN –website, last modified 04.04.2011, used 8.1.2015, downloadable from http://www.tumosan.com.tr/tr/images/traktor/8100/8195.jpg

[21] SolidWorks –website, http://www.solidworks.com/, internet source, last modified 13.11.2014, used 13.11.2014.

[22] Blender –website, http://www.blender.org/, internet source, last modified 13.11.2014, used 13.11.2014.

[23] Python –website, https://www.python.org/, internet source, last modified 13.11.2014, used 13.11.2014.

[24] Simulink, MathWorks –website, http://www.mathworks.se/products/simulink/, internet source, last modified 13.11.2014, used 13.11.2014.

[25] EconStats –website, http://www.econstats.com/wdi/wdiv___1.htm, internet source, used 4.11.2014

[26] Whitman, R. B., 1920, Tractor principles: The action, mechanism, handling, care, maintenance and repair of the gas engine tractor, D. Appleton and company, 292 p.

[27] Teinilä, T., 2009, Traktoreiden vaihteistot ja niiden tulevaisuus, Master's thesis, University of Helsinki, 101 p.

[28] Electric drivetrain –picture, Visedo –website, last modified 13.11.2013, used 19.01.2015, downloadable from http://www.visedo.com/images/stories/products/electric-drivetrains/1-visedo_hybrid_wheel_loader_light_www.jpg

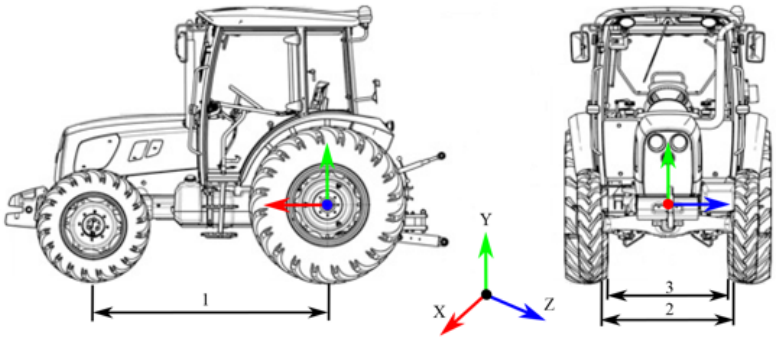
[29] MeVEA Tutorial 1, 30 p., last modified 10.01.2011.

APPENDICES

Appendix 1: Excel datasheet

Choose model properties

- Environment	Dome
- Body	Valtra
- Wheel type	Valtra_T202
Amount of wheels	
- At front	2
- At rear	2
- Motor	Valtra_T202
- Drive	2WD_1R



Origo is at the centre of rear axle.

Give following values

Distance between axles (1)	2748	mm		
Distance between front tyres (2)	1830	mm		
Distance between rear tyres (3)	1810	mm		
Front tyres:				
- Diameter	1168	mm		
- Width	500	mm		
- Mass	195	kg		
- Inertia	23.2	0	0	
		23.2	0	
			44.7	
Rear tyres:				
- Diameter	1700	mm		
- Width	500	mm		
- Mass	330	kg		
- Inertia	70	0	0	
		70	0	
			133	
Cabin				
- Mass	800	kg		
- Location (x, y, z)	0.1	0.3	0	m
- Inertia	380	0	0	
		440	0	
			440	
Motor 1				
- Mass	200	kg		
- Location (x, y, z)	0.9	0.1	0	m
- Inertia	23	0	0	
		23	0	
			44	

(Continues)

Appendix 1: (Continued)

Motor 2				
- Mass	200	kg		
- Location (x, y, z)	0.9	0.1	0	m
- Inertia	23	0	0	
		23	0	
			44	
Motor 3				
- Mass	200	kg		
- Location (x, y, z)	0.9	0.1	0	m
- Inertia	23	0	0	
		23	0	
			44	
Motor 4				
- Mass	200	kg		
- Location (x, y, z)	0.9	0.1	0	m
- Inertia	23	0	0	
		23	0	
			44	
Fueltank				
- Mass	200	kg		
- Location (x, y, z)	0.7	0	0.1	m
- Inertia	23	0	0	
		23	0	
			44	
Extra load at front				
- Mass	1	kg		
- Location (x, y, z)	1	0	0	m
- Inertia	1	0	0	
		1	0	
			1	
Extra load at rear				
- Mass	1	kg		
- Location (x, y, z)	-0.1	0	0	m
- Inertia	1	0	0	
		1	0	
			1	
Batteries				
- Mass	80	kg		
- Location (x, y, z)	0.05	0.1	0	m
- Inertia	10	0	0	
		10	0	
			10	
Generator				
- Mass	150	kg		
- Location (x, y, z)	1	0.1	0	m

(Continues)

Appendix 1: (Continued)

- Inertia	23	0	0	
		23	0	
			44	
Efficiency of powertransmission			0.85	
Equipments	Trailer			
- Mass of the Load	5000	kg		
- Inertia	500	0	0	
		500	0	
			500	
Converter 1				
- Mass	10	kg		
- Location (x, y, z)	0	0	0	
- Inertia	23	0	0	
		23	0	
			44	
Converter 2				
- Mass	10	kg		
- Location (x, y, z)	0	0	0	
- Inertia	23	0	0	
		23	0	
			44	
Converter 3				
- Mass	10	kg		
- Location (x, y, z)	0	0	0	
- Inertia	23	0	0	
		23	0	
			44	
Converter 4				
- Mass	10	kg		
- Location (x, y, z)	0	0	0	
- Inertia	23	0	0	
		23	0	
			44	
Converter 5				
- Mass	10	kg		
- Location (x, y, z)	0	0	0	
- Inertia	23	0	0	
		23	0	
			44	
Maximum speed	50	km/h		

Appendix 2: Python –script

```
1
2 # Importing nesessary moduls
3 from xlrd import open_workbook
4 import os
5
6
7
8 # Funktions which will be used later.
9
10
11 def basic_replace(content, searchable, replacement):
12
13     contentOld = content
14     content = []
15
16     for line in contentOld:
17         line = line.replace(searchable, replacement)
18         content.append(line)
19     return content
20
21 def make_assembly(BodyType, searcheable, replacement):
22
23     mva_handle = open('.\\Assemblies\\' + BodyType.replace('_Main','_base'),\
24     'r')
25     mva_content = mva_handle.readlines()
26     mva_handle.close()
27
28     mva_contentOld = mva_content
29     mva_content = []
30
31     for line in mva_contentOld:
32         line = line.replace(searcheable, replacement)
33         mva_content.append(line)
34
35     filu_mva = open('.\\Assemblies\\' + BodyType, 'w')
36     filu_mva.writelines(mva_content)
37     filu_mva.close()
38
39 def assembly_replace(BodyType, searcheable, replacement):
40
41     mva_handle = open('.\\Assemblies\\' + BodyType, 'r')
42     mva_content = mva_handle.readlines()
43     mva_handle.close()
44
45     mva_contentOld = mva_content
46     mva_content = []
47
48     for line in mva_contentOld:
49         line = line.replace(searcheable, replacement)
50         mva_content.append(line)
```

(Continues)

Appendix 2: (Continued)

```
51
52      filu_mva = open('.\\Assemblies\\' + BodyType, 'w')
53      filu_mva.writelines(mva_content)
54      filu_mva.close()
55
56  def make_assembly2(BodyType, searcheable, replacement):
57
58      mva_handle = open(BodyType.replace('_Main','_base'), 'r')
59      mva_content = mva_handle.readlines()
60      mva_handle.close()
61
62      mva_contentOld = mva_content
63      mva_content = []
64
65      for line in mva_contentOld:
66          line = line.replace(searcheable, replacement)
67          mva_content.append(line)
68
69      filu_mva = open(BodyType, 'w')
70      filu_mva.writelines(mva_content)
71      filu_mva.close()
72
73
74
75  book = open_workbook('ProductList.xlsx')
76  sheet = book.sheet_by_index(0)
77
78
79
80  #-----Wheels-----
81
82
83  WheelMass = str(float(sheet.cell(25,3).value) + 0.001)
84  BodyType = 'TyreSet_' + str(sheet.cell(7,2).value) + '_Main.mva'
85  make_assembly(BodyType, 'Replace_TyreF_mass', WheelMass)
86
87  WheelInertia = str(float(sheet.cell(27,3).value))
88  assembly_replace(BodyType, 'Replace_TyreF_Inertia_xx', WheelInertia)
89
90  WheelInertia = str(float(sheet.cell(28,4).value))
91  assembly_replace(BodyType, 'Replace_TyreF_Inertia_yy', WheelInertia)
92
93  WheelInertia = str(float(sheet.cell(29,5).value))
94  assembly_replace(BodyType, 'Replace_TyreF_Inertia_zz', WheelInertia)
95
96  WheelInertia = str(float(sheet.cell(27,4).value))
97  assembly_replace(BodyType, 'Replace_TyreF_Inertia_xy', WheelInertia)
98
99  WheelInertia = str(float(sheet.cell(27,5).value))
100 assembly_replace(BodyType, 'Replace_TyreF_Inertia_zx', WheelInertia)
```

(Continues)

Appendix 2: (Continued)

```
101
102 WheelInertia = str(float(sheet.cell(28,5).value))
103 assembly_replace(BodyType, 'Replace_TyreF_Inertia_yz', WheelInertia)
104
105
106
107 WheelMass = str(float(sheet.cell(33,3).value) + 0.001)
108 assembly_replace(BodyType, 'Replace_TyreR_mass', WheelMass)
109
110 WheelInertia = str(float(sheet.cell(35,3).value))
111 assembly_replace(BodyType, 'Replace_TyreR_Inertia_xx', WheelInertia)
112
113 WheelInertia = str(float(sheet.cell(36,4).value))
114 assembly_replace(BodyType, 'Replace_TyreR_Inertia_yy', WheelInertia)
115
116 WheelInertia = str(float(sheet.cell(37,5).value))
117 assembly_replace(BodyType, 'Replace_TyreR_Inertia_zz', WheelInertia)
118
119 WheelInertia = str(float(sheet.cell(35,4).value))
120 assembly_replace(BodyType, 'Replace_TyreR_Inertia_xy', WheelInertia)
121
122 WheelInertia = str(float(sheet.cell(35,5).value))
123 assembly_replace(BodyType, 'Replace_TyreR_Inertia_zx', WheelInertia)
124
125 WheelInertia = str(float(sheet.cell(36,5).value))
126 assembly_replace(BodyType, 'Replace_TyreR_Inertia_yz', WheelInertia)
127
128
129
130 FrontAxle_y = str((float(sheet.cell(23,3).value)-float(sheet.cell(31,3).\
131 value))/2000+0.05)
132 assembly_replace(BodyType, 'Replace_FrontAxle_y', FrontAxle_y)
133
134 FrontAxle_x = str(float(sheet.cell(19,3).value)/1000)
135 assembly_replace(BodyType, 'Replace_FrontAxle_x', FrontAxle_x)
136
137 BreadthFront = str(float(sheet.cell(20,3).value)/2000)
138 assembly_replace(BodyType, 'Replace_FrontBreadth', BreadthFront)
139
140 BreadthRear = str(float(sheet.cell(21,3).value)/2000)
141 assembly_replace(BodyType, 'Replace_RearBreadth', BreadthRear)
142
143 Spline_y_Rear = str(float(sheet.cell(31,3).value)/2000)
144 assembly_replace(BodyType, 'Replace_Spline_y_Rear', Spline_y_Rear)
145
146 Spline_x_Rear = str(float(sheet.cell(32,3).value)/2000)
147 assembly_replace(BodyType, 'Replace_Spline_x_Rear', Spline_x_Rear)
148
149 Spline_y_Front = str(float(sheet.cell(23,3).value)/2000)
150 assembly_replace(BodyType, 'Replace_Spline_y_Front', Spline_y_Front)
```

(Continues)

Appendix 2: (Continued)

```
151
152 Spline_x_Front = str(float(sheet.cell(24,3).value)/2000)
153 assembly_replace(BodyType, 'Replace_Spline_x_Front', Spline_x_Front)
154
155
156
157 WheelMass = str(float(sheet.cell(25,3).value) + 0.001)
158 BodyType = 'TyreSet_' + str(sheet.cell(7,2).value) + '_4xFront_Main.mva'
159 make_assembly(BodyType, 'Replace_TyreF_mass', WheelMass)
160
161 WheelInertia = str(float(sheet.cell(27,3).value))
162 assembly_replace(BodyType, 'Replace_TyreF_Inertia_xx', WheelInertia)
163
164 WheelInertia = str(float(sheet.cell(28,4).value))
165 assembly_replace(BodyType, 'Replace_TyreF_Inertia_yy', WheelInertia)
166
167 WheelInertia = str(float(sheet.cell(29,5).value))
168 assembly_replace(BodyType, 'Replace_TyreF_Inertia_zz', WheelInertia)
169
170 WheelInertia = str(float(sheet.cell(27,4).value))
171 assembly_replace(BodyType, 'Replace_TyreF_Inertia_xy', WheelInertia)
172
173 WheelInertia = str(float(sheet.cell(27,5).value))
174 assembly_replace(BodyType, 'Replace_TyreF_Inertia_zx', WheelInertia)
175
176 WheelInertia = str(float(sheet.cell(28,5).value))
177 assembly_replace(BodyType, 'Replace_TyreF_Inertia_yz', WheelInertia)
178
179 Wheel2Location = str(float(sheet.cell(24,3).value)/1000+0.03)
180 assembly_replace(BodyType, 'Replace_TyreF2_Location', Wheel2Location)
181
182
183
184 WheelMass = str(float(sheet.cell(33,3).value) + 0.001)
185 BodyType = 'TyreSet_' + str(sheet.cell(7,2).value) + '_4xRear_Main.mva'
186 make_assembly(BodyType, 'Replace_TyreR_mass', WheelMass)
187
188 WheelInertia = str(float(sheet.cell(35,3).value))
189 assembly_replace(BodyType, 'Replace_TyreR_Inertia_xx', WheelInertia)
190
191 WheelInertia = str(float(sheet.cell(36,4).value))
192 assembly_replace(BodyType, 'Replace_TyreR_Inertia_yy', WheelInertia)
193
194 WheelInertia = str(float(sheet.cell(37,5).value))
195 assembly_replace(BodyType, 'Replace_TyreR_Inertia_zz', WheelInertia)
196
197 WheelInertia = str(float(sheet.cell(35,4).value))
198 assembly_replace(BodyType, 'Replace_TyreR_Inertia_xy', WheelInertia)
199
200 WheelInertia = str(float(sheet.cell(35,5).value))
```

(Continues)

Appendix 2: (Continued)

```
assembly_replace(BodyType, 'Replace_TyreR_Inertia_zx', WheelInertia)

WheelInertia = str(float(sheet.cell(36,5).value))
assembly_replace(BodyType, 'Replace_TyreR_Inertia_yz', WheelInertia)

Wheel2Location = str(float(sheet.cell(32,3).value)*1.5/1000+0.02+float(sheet.\
cell(21,3).value)/2000)
assembly_replace(BodyType, 'Replace_TyreR2_Location', Wheel2Location)



#-----Body-----


CabinMass = str(float(sheet.cell(39,3).value) + 0.001)
BodyType = 'Cabin_' + str(sheet.cell(5,2).value) + '_Main.mva'
make_assembly(BodyType, 'Replace_Cabin_mass', CabinMass)

CabinInertia = str(float(sheet.cell(42,3).value))
assembly_replace(BodyType, 'Replace_Cabin_Inertia_xx', CabinInertia)

CabinInertia = str(float(sheet.cell(43,4).value))
assembly_replace(BodyType, 'Replace_Cabin_Inertia_yy', CabinInertia)

CabinInertia = str(float(sheet.cell(44,5).value))
assembly_replace(BodyType, 'Replace_Cabin_Inertia_zz', CabinInertia)

CabinInertia = str(float(sheet.cell(42,4).value))
assembly_replace(BodyType, 'Replace_Cabin_Inertia_xy', CabinInertia)

CabinInertia = str(float(sheet.cell(42,5).value))
assembly_replace(BodyType, 'Replace_Cabin_Inertia_zx', CabinInertia)

CabinInertia = str(float(sheet.cell(43,5).value))
assembly_replace(BodyType, 'Replace_Cabin_Inertia_yz', CabinInertia)

CabinLocation = str(float(sheet.cell(40,3).value))
assembly_replace(BodyType, 'Replace_Cabin_Location_x', CabinLocation)

CabinLocation = str(float(sheet.cell(40,4).value))
assembly_replace(BodyType, 'Replace_Cabin_Location_y', CabinLocation)

CabinLocation = str(float(sheet.cell(40,5).value))
assembly_replace(BodyType, 'Replace_Cabin_Location_z', CabinLocation)


FrameHeight = str(float(sheet.cell(31,3).value)/2000)
assembly_replace(BodyType, 'Replace_Frame_Height', FrameHeight)


```

(Continues)

Appendix 2: (Continued)

```
#-----Extra Dummies-----


Motor1Mass = str(float(sheet.cell(46,3).value) + 0.001)
BodyType = 'Extra_Dummies_Main.mva'
make_assembly(BodyType, 'Replace_Motor1_mass', Motor1Mass)

Motor1Location = str(float(sheet.cell(47,3).value))
assembly_replace(BodyType, 'Replace_Motor1_Location_x', Motor1Location)

Motor1Location = str(float(sheet.cell(47,4).value))
assembly_replace(BodyType, 'Replace_Motor1_Location_y', Motor1Location)

Motor1Location = str(float(sheet.cell(47,5).value))
assembly_replace(BodyType, 'Replace_Motor1_Location_z', Motor1Location)

Motor1Inertia = str(float(sheet.cell(49,3).value))
assembly_replace(BodyType, 'Replace_Motor1_Inertia_xx', Motor1Inertia)

Motor1Inertia = str(float(sheet.cell(50,4).value))
assembly_replace(BodyType, 'Replace_Motor1_Inertia_yy', Motor1Inertia)

Motor1Inertia = str(float(sheet.cell(51,5).value))
assembly_replace(BodyType, 'Replace_Motor1_Inertia_zz', Motor1Inertia)

Motor1Inertia = str(float(sheet.cell(49,4).value))
assembly_replace(BodyType, 'Replace_Motor1_Inertia_xy', Motor1Inertia)

Motor1Inertia = str(float(sheet.cell(49,5).value))
assembly_replace(BodyType, 'Replace_Motor1_Inertia_zx', Motor1Inertia)

Motor1Inertia = str(float(sheet.cell(50,5).value))
assembly_replace(BodyType, 'Replace_Motor1_Inertia_yz', Motor1Inertia)



Motor2Mass = str(float(sheet.cell(53,3).value) + 0.001)
assembly_replace(BodyType, 'Replace_Motor2_mass', Motor2Mass)

Motor2Location = str(float(sheet.cell(54,3).value))
assembly_replace(BodyType, 'Replace_Motor2_Location_x', Motor2Location)

Motor2Location = str(float(sheet.cell(54,4).value))
assembly_replace(BodyType, 'Replace_Motor2_Location_y', Motor2Location)

Motor2Location = str(float(sheet.cell(54,5).value))
assembly_replace(BodyType, 'Replace_Motor2_Location_z', Motor2Location)

Motor2Inertia = str(float(sheet.cell(56,3).value))
```

(Continues)

Appendix 2: (Continued)

```
assembly_replace(BodyType, 'Replace_Motor2_Inertia_xx', Motor2Inertia)

Motor2Inertia = str(float(sheet.cell(57,4).value))
assembly_replace(BodyType, 'Replace_Motor2_Inertia_yy', Motor2Inertia)

Motor2Inertia = str(float(sheet.cell(58,5).value))
assembly_replace(BodyType, 'Replace_Motor2_Inertia_zz', Motor2Inertia)

Motor2Inertia = str(float(sheet.cell(56,4).value))
assembly_replace(BodyType, 'Replace_Motor2_Inertia_xy', Motor2Inertia)

Motor2Inertia = str(float(sheet.cell(56,5).value))
assembly_replace(BodyType, 'Replace_Motor2_Inertia_zx', Motor2Inertia)

Motor2Inertia = str(float(sheet.cell(57,5).value))
assembly_replace(BodyType, 'Replace_Motor2_Inertia_yz', Motor2Inertia)



Motor3Mass = str(float(sheet.cell(60,3).value) + 0.001)
assembly_replace(BodyType, 'Replace_Motor3_mass', Motor3Mass)

Motor3Location = str(float(sheet.cell(61,3).value))
assembly_replace(BodyType, 'Replace_Motor3_Location_x', Motor3Location)

Motor3Location = str(float(sheet.cell(61,4).value))
assembly_replace(BodyType, 'Replace_Motor3_Location_y', Motor3Location)

Motor3Location = str(float(sheet.cell(61,5).value))
assembly_replace(BodyType, 'Replace_Motor3_Location_z', Motor3Location)

Motor3Inertia = str(float(sheet.cell(63,3).value))
assembly_replace(BodyType, 'Replace_Motor3_Inertia_xx', Motor3Inertia)

Motor3Inertia = str(float(sheet.cell(64,4).value))
assembly_replace(BodyType, 'Replace_Motor3_Inertia_yy', Motor3Inertia)

Motor3Inertia = str(float(sheet.cell(65,5).value))
assembly_replace(BodyType, 'Replace_Motor3_Inertia_zz', Motor3Inertia)

Motor3Inertia = str(float(sheet.cell(63,4).value))
assembly_replace(BodyType, 'Replace_Motor3_Inertia_xy', Motor3Inertia)

Motor3Inertia = str(float(sheet.cell(63,5).value))
assembly_replace(BodyType, 'Replace_Motor3_Inertia_zx', Motor3Inertia)

Motor3Inertia = str(float(sheet.cell(64,5).value))
assembly_replace(BodyType, 'Replace_Motor3_Inertia_yz', Motor3Inertia)
Motor2Mass = str(float(sheet.cell(53,3).value) + 0.001)
assembly_replace(BodyType, 'Replace_Motor2_mass', Motor2Mass)
```

(Continues)

Appendix 2: (Continued)

```
351
352 Motor2Location = str(float(sheet.cell(54,3).value))
353 assembly_replace(BodyType, 'Replace_Motor2_Location_x', Motor2Location)
354
355 Motor2Location = str(float(sheet.cell(54,4).value))
356 assembly_replace(BodyType, 'Replace_Motor2_Location_y', Motor2Location)
357
358 Motor2Location = str(float(sheet.cell(54,5).value))
359 assembly_replace(BodyType, 'Replace_Motor2_Location_z', Motor2Location)
360
361 Motor2Inertia = str(float(sheet.cell(56,3).value))
362 assembly_replace(BodyType, 'Replace_Motor2_Inertia_xx', Motor2Inertia)
363
364 Motor2Inertia = str(float(sheet.cell(57,4).value))
365 assembly_replace(BodyType, 'Replace_Motor2_Inertia_yy', Motor2Inertia)
366
367 Motor2Inertia = str(float(sheet.cell(58,5).value))
368 assembly_replace(BodyType, 'Replace_Motor2_Inertia_zz', Motor2Inertia)
369
370 Motor2Inertia = str(float(sheet.cell(56,4).value))
371 assembly_replace(BodyType, 'Replace_Motor2_Inertia_xy', Motor2Inertia)
372
373 Motor2Inertia = str(float(sheet.cell(56,5).value))
374 assembly_replace(BodyType, 'Replace_Motor2_Inertia_zx', Motor2Inertia)
375
376 Motor2Inertia = str(float(sheet.cell(57,5).value))
377 assembly_replace(BodyType, 'Replace_Motor2_Inertia_yz', Motor2Inertia)
378
379
380 Motor4Mass = str(float(sheet.cell(67,3).value) + 0.001)
381 assembly_replace(BodyType, 'Replace_Motor4_mass', Motor4Mass)
382
383 Motor4Location = str(float(sheet.cell(68,3).value))
384 assembly_replace(BodyType, 'Replace_Motor4_Location_x', Motor4Location)
385
386 Motor4Location = str(float(sheet.cell(68,4).value))
387 assembly_replace(BodyType, 'Replace_Motor4_Location_y', Motor4Location)
388
389 Motor4Location = str(float(sheet.cell(68,5).value))
390 assembly_replace(BodyType, 'Replace_Motor4_Location_z', Motor4Location)
391
392 Motor4Inertia = str(float(sheet.cell(70,3).value))
393 assembly_replace(BodyType, 'Replace_Motor4_Inertia_xx', Motor4Inertia)
394
395 Motor4Inertia = str(float(sheet.cell(71,4).value))
396 assembly_replace(BodyType, 'Replace_Motor4_Inertia_yy', Motor4Inertia)
397
398 Motor4Inertia = str(float(sheet.cell(72,5).value))
399 assembly_replace(BodyType, 'Replace_Motor4_Inertia_zz', Motor4Inertia)
400
```

(Continues)

Appendix 2: (Continued)

```
Motor4Inertia = str(float(sheet.cell(70,4).value))
assembly_replace(BodyType, 'Replace_Motor4_Inertia_xy', Motor4Inertia)

Motor4Inertia = str(float(sheet.cell(70,5).value))
assembly_replace(BodyType, 'Replace_Motor4_Inertia_zx', Motor4Inertia)

Motor4Inertia = str(float(sheet.cell(71,5).value))
assembly_replace(BodyType, 'Replace_Motor4_Inertia_yz', Motor4Inertia)



FueltankMass = str(float(sheet.cell(74,3).value) + 0.001)
assembly_replace(BodyType, 'Replace_Fueltank_mass', FueltankMass)

FueltankLocation = str(float(sheet.cell(75,3).value))
assembly_replace(BodyType, 'Replace_Fueltank_Location_x', FueltankLocation)

FueltankLocation = str(float(sheet.cell(75,4).value))
assembly_replace(BodyType, 'Replace_Fueltank_Location_y', FueltankLocation)

FueltankLocation = str(float(sheet.cell(75,5).value))
assembly_replace(BodyType, 'Replace_Fueltank_Location_z', FueltankLocation)

FueltankInertia = str(float(sheet.cell(77,3).value))
assembly_replace(BodyType, 'Replace_Fueltank_Inertia_xx', FueltankInertia)

FueltankInertia = str(float(sheet.cell(78,4).value))
assembly_replace(BodyType, 'Replace_Fueltank_Inertia_yy', FueltankInertia)

FueltankInertia = str(float(sheet.cell(79,5).value))
assembly_replace(BodyType, 'Replace_Fueltank_Inertia_zz', FueltankInertia)

FueltankInertia = str(float(sheet.cell(77,4).value))
assembly_replace(BodyType, 'Replace_Fueltank_Inertia_xy', FueltankInertia)

FueltankInertia = str(float(sheet.cell(77,5).value))
assembly_replace(BodyType, 'Replace_Fueltank_Inertia_zx', FueltankInertia)

FueltankInertia = str(float(sheet.cell(78,5).value))
assembly_replace(BodyType, 'Replace_Fueltank_Inertia_yz', FueltankInertia)



FrontLoadMass = str(float(sheet.cell(81,3).value) + 0.001)
assembly_replace(BodyType, 'Replace_FrontLoad_mass', FrontLoadMass)

FrontLoadLocation = str(float(sheet.cell(82,3).value))
assembly_replace(BodyType, 'Replace_FrontLoad_Location_x', FrontLoadLocation)

FrontLoadLocation = str(float(sheet.cell(82,4).value))
```

(Continues)

Appendix 2: (Continued)

```
assembly_replace(BodyType, 'Replace_FrontLoad_Location_y', FrontLoadLocation)

FrontLoadLocation = str(float(sheet.cell(82,5).value))
assembly_replace(BodyType, 'Replace_FrontLoad_Location_z', FrontLoadLocation)

FrontLoadInertia = str(float(sheet.cell(84,3).value))
assembly_replace(BodyType, 'Replace_FrontLoad_Inertia_xx', FrontLoadInertia)

FrontLoadInertia = str(float(sheet.cell(85,4).value))
assembly_replace(BodyType, 'Replace_FrontLoad_Inertia_yy', FrontLoadInertia)

FrontLoadInertia = str(float(sheet.cell(86,5).value))
assembly_replace(BodyType, 'Replace_FrontLoad_Inertia_zz', FrontLoadInertia)

FrontLoadInertia = str(float(sheet.cell(84,4).value))
assembly_replace(BodyType, 'Replace_FrontLoad_Inertia_xy', FrontLoadInertia)

FrontLoadInertia = str(float(sheet.cell(84,5).value))
assembly_replace(BodyType, 'Replace_FrontLoad_Inertia_zx', FrontLoadInertia)

FrontLoadInertia = str(float(sheet.cell(85,5).value))
assembly_replace(BodyType, 'Replace_FrontLoad_Inertia_yz', FrontLoadInertia)



RearLoadMass = str(float(sheet.cell(88,3).value) + 0.001)
assembly_replace(BodyType, 'Replace_RearLoad_mass', RearLoadMass)

RearLoadLocation = str(float(sheet.cell(89,3).value))
assembly_replace(BodyType, 'Replace_RearLoad_Location_x', RearLoadLocation)

RearLoadLocation = str(float(sheet.cell(89,4).value))
assembly_replace(BodyType, 'Replace_RearLoad_Location_y', RearLoadLocation)

RearLoadLocation = str(float(sheet.cell(89,5).value))
assembly_replace(BodyType, 'Replace_RearLoad_Location_z', RearLoadLocation)

RearLoadInertia = str(float(sheet.cell(91,3).value))
assembly_replace(BodyType, 'Replace_RearLoad_Inertia_xx', RearLoadInertia)

RearLoadInertia = str(float(sheet.cell(92,4).value))
assembly_replace(BodyType, 'Replace_RearLoad_Inertia_yy', RearLoadInertia)

RearLoadInertia = str(float(sheet.cell(93,5).value))
assembly_replace(BodyType, 'Replace_RearLoad_Inertia_zz', RearLoadInertia)

RearLoadInertia = str(float(sheet.cell(91,4).value))
assembly_replace(BodyType, 'Replace_RearLoad_Inertia_xy', RearLoadInertia)

RearLoadInertia = str(float(sheet.cell(91,5).value))
```

(Continues)

Appendix 2: (Continued)

```
assembly_replace(BodyType, 'Replace_RearLoad_Inertia_zx', RearLoadInertia)

RearLoadInertia = str(float(sheet.cell(92,5).value))
assembly_replace(BodyType, 'Replace_RearLoad_Inertia_yz', RearLoadInertia)



BatteriesMass = str(float(sheet.cell(95,3).value) + 0.001)
assembly_replace(BodyType, 'Replace_Batteries_mass', BatteriesMass)

BatteriesLocation = str(float(sheet.cell(96,3).value))
assembly_replace(BodyType, 'Replace_Batteries_Location_x', BatteriesLocation)

BatteriesLocation = str(float(sheet.cell(96,4).value))
assembly_replace(BodyType, 'Replace_Batteries_Location_y', BatteriesLocation)

BatteriesLocation = str(float(sheet.cell(96,5).value))
assembly_replace(BodyType, 'Replace_Batteries_Location_z', BatteriesLocation)

BatteriesInertia = str(float(sheet.cell(98,3).value))
assembly_replace(BodyType, 'Replace_Batteries_Inertia_xx', BatteriesInertia)

BatteriesInertia = str(float(sheet.cell(99,4).value))
assembly_replace(BodyType, 'Replace_Batteries_Inertia_yy', BatteriesInertia)

BatteriesInertia = str(float(sheet.cell(100,5).value))
assembly_replace(BodyType, 'Replace_Batteries_Inertia_zz', BatteriesInertia)

BatteriesInertia = str(float(sheet.cell(98,4).value))
assembly_replace(BodyType, 'Replace_Batteries_Inertia_xy', BatteriesInertia)

BatteriesInertia = str(float(sheet.cell(98,5).value))
assembly_replace(BodyType, 'Replace_Batteries_Inertia_zx', BatteriesInertia)

BatteriesInertia = str(float(sheet.cell(99,5).value))
assembly_replace(BodyType, 'Replace_Batteries_Inertia_yz', BatteriesInertia)



GeneratorMass = str(float(sheet.cell(102,3).value) + 0.001)
assembly_replace(BodyType, 'Replace_Generator_mass', GeneratorMass)

GeneratorLocation = str(float(sheet.cell(103,3).value))
assembly_replace(BodyType, 'Replace_Generator_Location_x', GeneratorLocation)

GeneratorLocation = str(float(sheet.cell(103,4).value))
assembly_replace(BodyType, 'Replace_Generator_Location_y', GeneratorLocation)

GeneratorLocation = str(float(sheet.cell(103,5).value))
assembly_replace(BodyType, 'Replace_Generator_Location_z', GeneratorLocation)
```

(Continues)

Appendix 2: (Continued)

```
551
552 GeneratorInertia = str(float(sheet.cell(105,3).value))
553 assembly_replace(BodyType, 'Replace_Generator_Inertia_xx', GeneratorInertia)
554
555 GeneratorInertia = str(float(sheet.cell(106,4).value))
556 assembly_replace(BodyType, 'Replace_Generator_Inertia_yy', GeneratorInertia)
557
558 GeneratorInertia = str(float(sheet.cell(107,5).value))
559 assembly_replace(BodyType, 'Replace_Generator_Inertia_zz', GeneratorInertia)
560
561 GeneratorInertia = str(float(sheet.cell(105,4).value))
562 assembly_replace(BodyType, 'Replace_Generator_Inertia_xy', GeneratorInertia)
563
564 GeneratorInertia = str(float(sheet.cell(105,5).value))
565 assembly_replace(BodyType, 'Replace_Generator_Inertia_zx', GeneratorInertia)
566
567 GeneratorInertia = str(float(sheet.cell(106,5).value))
568 assembly_replace(BodyType, 'Replace_Generator_Inertia_yz', GeneratorInertia)
569
570
571
572 Converter1Mass = str(float(sheet.cell(119,3).value) + 0.001)
573 assembly_replace(BodyType, 'Replace_Converter1_mass', Converter1Mass)
574
575 Converter1Location = str(float(sheet.cell(120,3).value))
576 assembly_replace(BodyType, 'Replace_Converter1_Location_x', Converter1Location)
577
578 Converter1Location = str(float(sheet.cell(120,4).value))
579 assembly_replace(BodyType, 'Replace_Converter1_Location_y', Converter1Location)
580
581 Converter1Location = str(float(sheet.cell(120,5).value))
582 assembly_replace(BodyType, 'Replace_Converter1_Location_z', Converter1Location)
583
584 Converter1Inertia = str(float(sheet.cell(122,3).value))
585 assembly_replace(BodyType, 'Replace_Converter1_Inertia_xx', Converter1Inertia)
586
587 Converter1Inertia = str(float(sheet.cell(123,4).value))
588 assembly_replace(BodyType, 'Replace_Converter1_Inertia_yy', Converter1Inertia)
589
590 Converter1Inertia = str(float(sheet.cell(124,5).value))
591 assembly_replace(BodyType, 'Replace_Converter1_Inertia_zz', Converter1Inertia)
592
593 Converter1Inertia = str(float(sheet.cell(122,4).value))
594 assembly_replace(BodyType, 'Replace_Converter1_Inertia_xy', Converter1Inertia)
595
596 Converter1Inertia = str(float(sheet.cell(122,5).value))
597 assembly_replace(BodyType, 'Replace_Converter1_Inertia_zx', Converter1Inertia)
598
599 Converter1Inertia = str(float(sheet.cell(123,5).value))
600 assembly_replace(BodyType, 'Replace_Converter1_Inertia_yz', Converter1Inertia)
```

(Continues)

Appendix 2: (Continued)

```
601
602
603
604 Converter2Mass = str(float(sheet.cell(126,3).value) + 0.001)
605 assembly_replace(BodyType, 'Replace_Converter2_mass', Converter2Mass)
606
607 Converter2Location = str(float(sheet.cell(127,3).value))
608 assembly_replace(BodyType, 'Replace_Converter2_Location_x', Converter2Location)
609
610 Converter2Location = str(float(sheet.cell(127,4).value))
611 assembly_replace(BodyType, 'Replace_Converter2_Location_y', Converter2Location)
612
613 Converter2Location = str(float(sheet.cell(127,5).value))
614 assembly_replace(BodyType, 'Replace_Converter2_Location_z', Converter2Location)
615
616 Converter2Inertia = str(float(sheet.cell(129,3).value))
617 assembly_replace(BodyType, 'Replace_Converter2_Inertia_xx', Converter2Inertia)
618
619 Converter2Inertia = str(float(sheet.cell(130,4).value))
620 assembly_replace(BodyType, 'Replace_Converter2_Inertia_yy', Converter2Inertia)
621
622 Converter2Inertia = str(float(sheet.cell(131,5).value))
623 assembly_replace(BodyType, 'Replace_Converter2_Inertia_zz', Converter2Inertia)
624
625 Converter2Inertia = str(float(sheet.cell(129,4).value))
626 assembly_replace(BodyType, 'Replace_Converter2_Inertia_xy', Converter2Inertia)
627
628 Converter2Inertia = str(float(sheet.cell(129,5).value))
629 assembly_replace(BodyType, 'Replace_Converter2_Inertia_zx', Converter2Inertia)
630
631 Converter2Inertia = str(float(sheet.cell(130,5).value))
632 assembly_replace(BodyType, 'Replace_Converter2_Inertia_yz', Converter2Inertia)
633
634
635
636 Converter3Mass = str(float(sheet.cell(133,3).value) + 0.001)
637 assembly_replace(BodyType, 'Replace_Converter3_mass', Converter3Mass)
638
639 Converter3Location = str(float(sheet.cell(134,3).value))
640 assembly_replace(BodyType, 'Replace_Converter3_Location_x', Converter3Location)
641
642 Converter3Location = str(float(sheet.cell(134,4).value))
643 assembly_replace(BodyType, 'Replace_Converter3_Location_y', Converter3Location)
644
645 Converter3Location = str(float(sheet.cell(134,5).value))
646 assembly_replace(BodyType, 'Replace_Converter3_Location_z', Converter3Location)
647
648 Converter3Inertia = str(float(sheet.cell(136,3).value))
649 assembly_replace(BodyType, 'Replace_Converter3_Inertia_xx', Converter3Inertia)
650
```

(Continues)

Appendix 2: (Continued)

```
Converter3Inertia = str(float(sheet.cell(137,4).value))
assembly_replace(BodyType, 'Replace_Converter3_Inertia_yy', Converter3Inertia)

Converter3Inertia = str(float(sheet.cell(138,5).value))
assembly_replace(BodyType, 'Replace_Converter3_Inertia_zz', Converter3Inertia)

Converter3Inertia = str(float(sheet.cell(136,4).value))
assembly_replace(BodyType, 'Replace_Converter3_Inertia_xy', Converter3Inertia)

Converter3Inertia = str(float(sheet.cell(136,5).value))
assembly_replace(BodyType, 'Replace_Converter3_Inertia_zx', Converter3Inertia)

Converter3Inertia = str(float(sheet.cell(137,5).value))
assembly_replace(BodyType, 'Replace_Converter3_Inertia_yz', Converter3Inertia)



Converter4Mass = str(float(sheet.cell(140,3).value) + 0.001)
assembly_replace(BodyType, 'Replace_Converter4_mass', Converter4Mass)

Converter4Location = str(float(sheet.cell(141,3).value))
assembly_replace(BodyType, 'Replace_Converter4_Location_x', Converter4Location)

Converter4Location = str(float(sheet.cell(141,4).value))
assembly_replace(BodyType, 'Replace_Converter4_Location_y', Converter4Location)

Converter4Location = str(float(sheet.cell(141,5).value))
assembly_replace(BodyType, 'Replace_Converter4_Location_z', Converter4Location)

Converter4Inertia = str(float(sheet.cell(143,3).value))
assembly_replace(BodyType, 'Replace_Converter4_Inertia_xx', Converter4Inertia)

Converter4Inertia = str(float(sheet.cell(144,4).value))
assembly_replace(BodyType, 'Replace_Converter4_Inertia_yy', Converter4Inertia)

Converter4Inertia = str(float(sheet.cell(145,5).value))
assembly_replace(BodyType, 'Replace_Converter4_Inertia_zz', Converter4Inertia)

Converter4Inertia = str(float(sheet.cell(143,4).value))
assembly_replace(BodyType, 'Replace_Converter4_Inertia_xy', Converter4Inertia)

Converter4Inertia = str(float(sheet.cell(143,5).value))
assembly_replace(BodyType, 'Replace_Converter4_Inertia_zx', Converter4Inertia)

Converter4Inertia = str(float(sheet.cell(144,5).value))
assembly_replace(BodyType, 'Replace_Converter4_Inertia_yz', Converter4Inertia)



Converter5Mass = str(float(sheet.cell(147,3).value) + 0.001)
```

(Continues)

Appendix 2: (Continued)

```
assembly_replace(BodyType, 'Replace_Converter5_mass', Converter5Mass)

Converter5Location = str(float(sheet.cell(148,3).value))
assembly_replace(BodyType, 'Replace_Converter5_Location_x', Converter5Location)

Converter5Location = str(float(sheet.cell(148,4).value))
assembly_replace(BodyType, 'Replace_Converter5_Location_y', Converter5Location)

Converter5Location = str(float(sheet.cell(148,5).value))
assembly_replace(BodyType, 'Replace_Converter5_Location_z', Converter5Location)

Converter5Inertia = str(float(sheet.cell(150,3).value))
assembly_replace(BodyType, 'Replace_Converter5_Inertia_xx', Converter5Inertia)

Converter5Inertia = str(float(sheet.cell(151,4).value))
assembly_replace(BodyType, 'Replace_Converter5_Inertia_yy', Converter5Inertia)

Converter5Inertia = str(float(sheet.cell(152,5).value))
assembly_replace(BodyType, 'Replace_Converter5_Inertia_zz', Converter5Inertia)

Converter5Inertia = str(float(sheet.cell(150,4).value))
assembly_replace(BodyType, 'Replace_Converter5_Inertia_xy', Converter5Inertia)

Converter5Inertia = str(float(sheet.cell(150,5).value))
assembly_replace(BodyType, 'Replace_Converter5_Inertia_zx', Converter5Inertia)

Converter5Inertia = str(float(sheet.cell(151,5).value))
assembly_replace(BodyType, 'Replace_Converter5_Inertia_yz', Converter5Inertia)



# Building main generic model .xml file


MotorType = str(sheet.cell(13,2).value)
if MotorType == 'Valtra_T202':
    filename = 'Generic_model_0.xml'
else:
    ReadValue = str(sheet.cell(14,2).value)
    if ReadValue == '2WD_1R'or ReadValue == '2WD_1F' or ReadValue == '4WD_1':
        filename = 'Generic_model_1.xml'
    elif ReadValue == '2WD_2R' or ReadValue == '2WD_2F' or ReadValue == \
    '4WD_1R_1F':
        filename = 'Generic_model_2.xml'
    elif ReadValue == '4WD_1F_2R' or ReadValue == '4WD_2F_1R':
        filename = 'Generic_model_3.xml'
    elif ReadValue == '4WD_2R_2F':
        filename = 'Generic_model_4.xml'
    else:
        filename = 'Generic_model_0.xml'
```

(Continues)

Appendix 2: (Continued)

```
751
752
753 filu = open(filename, 'r')
754 content = filu.readlines()
755 filu.close()
756
757
758 BodyType = 'Environment_' + str(sheet.cell(3,2).value) + '_Main.mva'
759 content = basic_replace(content, 'Environment_Replace', BodyType)
760
761 BodyType = 'Cabin_' + str(sheet.cell(5,2).value) + '_Main.mva'
762 content = basic_replace(content, 'Body_Replace', BodyType)
763
764 BodyType = 'TyreSet_' + str(sheet.cell(7,2).value) + '_Main.mva'
765 content = basic_replace(content, 'TyreSet_Replace', BodyType)
766
767 ReadValue = float(sheet.cell(10,2).value)
768 if ReadValue == 4:
769     BodyType = 'TyreSet_' + str(sheet.cell(7,2).value) + '_4xFront_Main.mva'
770     content = basic_replace(content, 'TyreAmount_Front_Replace', BodyType)
771 else:
772     BodyType = 'EmptyFile.mva'
773     content = basic_replace(content, 'TyreAmount_Front_Replace', BodyType)
774
775 ReadValue = float(sheet.cell(11,2).value)
776 if ReadValue == 4:
777     BodyType = 'TyreSet_' + str(sheet.cell(7,2).value) + '_4xRear_Main.mva'
778     content = basic_replace(content, 'TyreAmount_Rear_Replace', BodyType)
779 else:
780     BodyType = 'EmptyFile.mva'
781     content = basic_replace(content, 'TyreAmount_Rear_Replace', BodyType)
782
783 BodyType = 'Transmission_' + str(sheet.cell(13,2).value) + '_' + str(sheet.\
784 cell(14,2).value) + '_Main.mva'
785 content = basic_replace(content, 'Transmission_Replace', BodyType)
786
787 ReadValue = str(sheet.cell(111,2).value)
788 if ReadValue == 'Trailer':
789     BodyType = 'AdditionalEquipment_Log_Trailer1_Main.mva'
790     content = basic_replace(content, 'Equipments_Replace1', BodyType)
791     BodyType = 'AdditionalEquipment_Log_Trailer2_Main.mva'
792     content = basic_replace(content, 'Equipments_Replace2', BodyType)
793
794     TrailerMass = str(float(sheet.cell(113,3).value) + 0.001)
795     BodyType = 'AdditionalEquipment_Log_Trailer2_Main.mva'
796     make_assembly(BodyType, 'Replace_TrailerLoad', TrailerMass)
797
798     TrailerInertia = str(float(sheet.cell(115,3).value))
799     assembly_replace(BodyType, 'Replace_TrailerInertia_xx', TrailerInertia)
800
```

(Continues)

Appendix 2: (Continued)

```
    TrailerInertia = str(float(sheet.cell(116,4).value))
    assembly_replace(BodyType, 'Replace_TrailerInertia_yy', TrailerInertia)

    TrailerInertia = str(float(sheet.cell(117,5).value))
    assembly_replace(BodyType, 'Replace_TrailerInertia_zz', TrailerInertia)

    TrailerInertia = str(float(sheet.cell(115,4).value))
    assembly_replace(BodyType, 'Replace_TrailerInertia_xy', TrailerInertia)

    TrailerInertia = str(float(sheet.cell(115,5).value))
    assembly_replace(BodyType, 'Replace_TrailerInertia_zx', TrailerInertia)

    TrailerInertia = str(float(sheet.cell(116,5).value))
    assembly_replace(BodyType, 'Replace_TrailerInertia_yz', TrailerInertia)

else:
    BodyType = 'EmptyFile.mva'
    content = basic_replace(content, 'Equipments_Replace1', BodyType)
    content = basic_replace(content, 'Equipments_Replace2', BodyType)

filename_new = filename[:-6] + '_base.xml'
filu = open(filename_new, 'w')
filu.writelines(content)
filu.close()

###END############ Building main generic model .xml file

BodyType = 'Generic_model_Main.xml'
Environment = str(sheet.cell(3,2).value)
if Environment == 'Dome':
    EnvironmentGraphics = 'Ground;Sky;Environment'
    make_assembly2(BodyType, 'Replace_Environment_Graphics', \
    EnvironmentGraphics)
else:
    EnvironmentGraphics = 'SandCavity_Graphics;VertexEdit_SandCavity_Graphics\
    ;SandCavityTracks_Graphics;Trees;W_stairs_Graphics;W_benchset_Graphics\
    ;C_pig_wall_Graphics;C_pig_slalom_Graphics;C_pig_uneven_Graphics\
    ;W_stumps_Graphics;C_gatepillars_Graphics;M_container_Graphics'
    make_assembly2(BodyType, 'Replace_Environment_Graphics', \
    EnvironmentGraphics)

    CollisionGraphics = 'COLL_SandCavity_Graphics;COLL_C_pig_wall_Graphics\
    ;COLL_C_pig_slalom_Graphics;COLL_C_pig_uneven_Graphics'
    basic_replace(BodyType, 'Replace_CollisionGraphics', CollisionGraphics)

print '-----  Script is ready!  -----'
```