Universität Bremen

Universität Bremen

Fachbereich 4: Produktionstechnik

Master Thesis zur Erlangung des akademischen Grades

Master of Science (M.Sc.)

Conception of an MBSE Workflow for the Execution of Thermal Analyses of Space Systems

Konzeptionierung eines MBSE Workflows zur Durchführung von Thermalanalysen von Raumfahrtsystemen

---

Vorgelegt von:
Henning Heibrok
Bornholmstraße 6
33729 Bielefeld
heibrok@uni-bremen.de

22. Februar 2021

Matrikelnummer: 4156897

Prüfer:

Prof. Dr. Claus Braxmaier

Dr.-Ing. Jens Grosse

Eidesstattliche Erklärung

Ich versichere hiermit, dass ich meine Masterarbeit "Konzeptionierung eines MBSE Workflows zur Durchführung von Thermalanalysen von Raumfahrtsystemen" selbstständig und ohne fremde Hilfe angefertigt und sie nicht vorher in einem anderen Prüfungsverfahren eingereicht habe und dass ich alle von anderen Autoren wörtlich übernommenen Stellen wie auch die sich an die Gedankengänge anderer Autoren eng anlehnenden Ausführungen meiner Arbeit besonders gekennzeichnet und die Quellen zitiert habe.

Bielefeld, den 22.02.2021

Heibrok

Henning Heibrok
2021.02.22 13:
25:56+01'00'

Henning Heibrok

Acknowledgments

First of all, I would like to express my gratitude to Dr. rer. nat. Stefan Schlüter, who enabled the opportunity of writing this thesis at Galileo Competence Center. Thanks is also given to my examiners Prof. Dr. Claus Braxmaier and Dr.-Ing Jens Grosse. I owe gratitude to Andrea Jara Garcia, Anton Donner, and Moritz Edelhäuser for always helping when questions arose. Furthermore I want to thank the whole Galileo Competence Center team for providing valuable ideas to improve this thesis.

Conception of an MBSE Workflow for the Execution of Thermal Analyses of Space Systems

DLR

Abstract

In this thesis a workflow was created, allowing to export the thermal model of a space system stored in the model-based systems engineering (MBSE) software Virtual Satellite and transition it into a thermal simulation model to conduct a thermal analysis using the finite element analysis (FEA) software CalculiX. To achieve this, first, a so called "Concept" was created in Virtual Satellite, enabling to store a thermal model inside the system model. In addition, a number of applications were written for Virtual Satellite to export the thermal model data from the system model. Furthermore a Python script was written for the computer aided design (CAD) software FreeCAD, that automatically meshes the geometry and assigns all thermal model data to the finite element mesh. The script then exports the complete thermal model into readable CalculiX input format. Moreover a function to include orbit data obtained using mission analysis tools was created. This orbit data is used to determine the solar radiation, Earth albedo, and Earth infrared radiation intensities over time and apply them as varying boundary conditions to dynamic simulations. For the interpretation of the simulation results, another application was written in Virtual Satellite. This application reads the CalculiX output file and assigns the obtained temperatures to the corresponding components. To check the validity of the workflow, a number of simulations were executed with ANSYS to compare the results to those of the workflow, obtained with CalculiX. The results showed very similar values, indicating that the workflow works as intended.

In der vorliegenden Thesis wurde ein Workflow erarbeitet, der es erlaubt, ein in der MBSE Software Virtual Satellite hinterlegtes Systemmodell eines Raumfahrtsystems zu exportieren und in ein Simulationsmodell zur Durchführung einer Thermalanalyse mit der FEA Software CalculiX zu überführen. Dafür wurde ein sogenanntes "Concept" in Virtual Satellite erstellt, um ein Thermalmodell innerhalb des Systemmodells speichern zu können. Weiterhin wurde eine Reihe von Applikationen in Virtual Satellite geschrieben, die den Export des Thermalmodells übernehmen. Für die CAD Software FreeCAD wurde weiterhin ein Python-Skript geschrieben, das das Finite-Elemente-Netz erzeugt und diesem die Daten des Thermalmodells zuweist. Anschließend exportiert das Skript alle Daten in das für CalculiX lesbare Input Format. Zusätzlich ist es möglich, dass das Skript mithilfe der Daten von Missionsanalysetools den Verlauf der Sonneneinstrahlung, Erdalbedo und Erdinfrarotstrahlung während des Orbits als Wärmelast auf den entsprechend orientierten Oberflächen einbindet. Für die Interpretation der Simulationsergebnisse wurde eine weitere Applikation für Virtual Satellite erstellt, die die Temperaturen der Komponenten aus der CalculiX Outputdatei liest und den jeweiligen Komponenten im Systemmodell zuweist. Um die Validität des Workflows zu prüfen wurden einige Vergleichssimulationen mit der etablierten FEA Software ANSYS durchgeführt, die sehr ähnliche Ergebnisse aufweisen und damit andeuten, dass der Workflow wie geplant funktioniert.

Contents

List of Figures

List of Tables

Acronyms

AOCS	attitude and orbit control subsystem
API	application programming interface
CAD	computer aided design
CAE	computer aided engineering
CDM	conceptual data model
CFD	computational fluid dynamics
CSV	comma-separated values
DLR	Deutsches Zentrum für Luft- und Raumfahrt e.V.
DLR-GK	DLR-Galileo Competence Center
DLR-SC	DLR-Institute of Simulation and Software Technology
ECSS	European Cooperation for Space Standardization
EEE	electrical, electronic and electromechanical
FDM	finite-differences method
FEA	finite element analysis
FEM	finite element methods
GMAT	General Mission Analysis Tool
GNSS	global navigation satellite system
GUI	graphical user interface
IDE	integrated development environment
INCOSE	International Council on Systems Engineering
ISS	International Space Station
JDK	Java Development Kit
JSON	JavaScript Object Notation
MBSE	model-based systems engineering
OCDT	Open Concurrent Design Tool
RAM	random-access memory
RCE	Remote Component Environment
SEBoK	Systems Engineering Body of Knowledge
SEI	structural element instance
STK	Systems Tool Kit
TCS	thermal control subsystem
TRL	technology readiness level
UML	unified modelling language
VCS	version control system
VirSat	Virtual Satellite
VSD	Virtual Spacecraft Design

Chapter 1

Introduction

Today, systems engineering is a widely spread practice in space industry [1]. Its tasks aim to handle complex systems with many components, dependencies and uncertainties and to ensure that the integrated system's capabilities eventually match the goals it was developed for [1]. The systems engineer's responsibilities during development begin in the earliest project phases and extend all the way to the acceptance review. They can be divided into three main process groups: System design processes, product realization processes and technical management processes [1]. The so called V-Model, as depicted in Fig. 1.1, gives an overview over the sequence and relationships of the top level tasks. It focuses mostly on system design and product realization processes. Activities in the left branch comprise, among other tasks, top level system design to bottom level component design, whereas the dominating part of the right branch are assembly, verification, integration and validation activities going from bottom level component verification to top level system validation. Being in the end of this sequence, it is indicated that the most important system level verifications and validations can only be conducted when the system is fully designed and integrated. This introduces a critical risk: If a design error leads to the failure of the system, a redesign is necessary. Depending on the severity of the error this can lead to the necessity to redesign other components as well, due to the high amount of dependencies between them. Having to redesign multiple subsystems essentially sets the project development back to the left branch of the V-Model. Consequently, this introduces substantial delays and, inevitably associated, a considerable increase in project cost.

This is where modeling comes into play. Models act as a short cut between the two branches of the V-Model so that verification and validation activities of the right branch can already be conducted in the left branch. The earlier an issue is detected, the less cost intensive it is to find a solution for it. This is why models are created in the first place. In a complex project numerous functional and physical models are generated from component level to system level, as early as possible [3]. They help the engineers to identify potential design issues without having to extensively test expensive hardware.

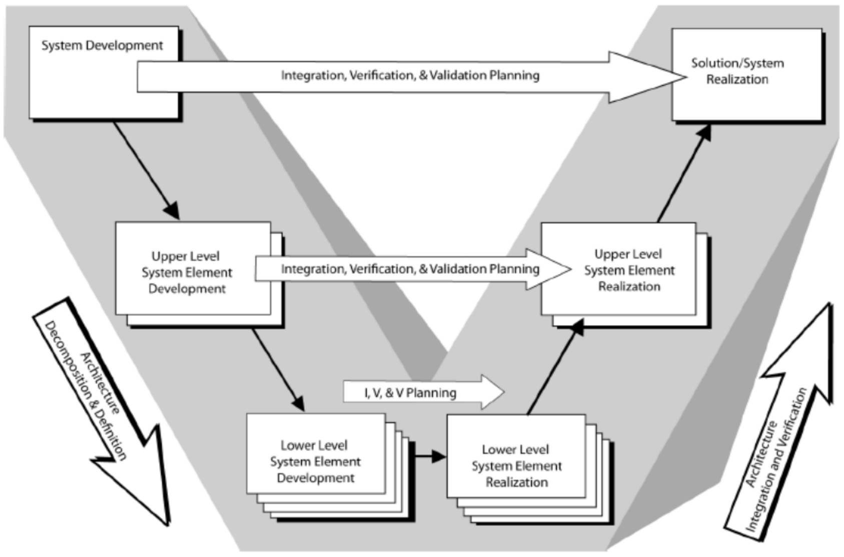

System Development
Integration, Verification, & Validation Planning
Solution/System Realization
Upper Level System Element Development
Integration, Verification, & Validation Planning
Upper Level System Element Realization
I, V, & V Planning
Lower Level System Element Development
Lower Level System Element Realization
Architecture Decomposition & Definition
Architecture Integration and Verification

Figure 1.1: V-Model of systems engineering. Source: Adopted from [2].

Table 1.1: Examples for the influences of an attitude and orbit control subsystem (AOCS) sensor change on other subsystems

Affected subsystem	Affected parameter
AOCS	Availability of attitude data, pointing accuracy
Data Handling	Dependability of attitude data
Payload	Pointing accuracy
Power	Power demand
Structure	Volume, fixation
System	Mass
Thermal	Operating temperature

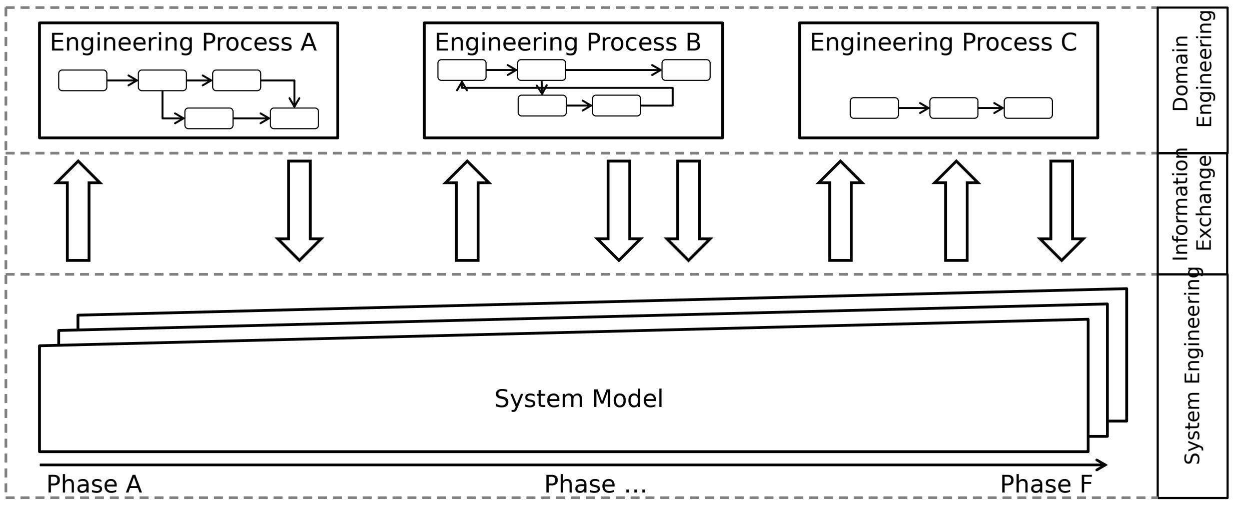


Figure 1.2: MBSE with one system model for all development phases. Source: Adopted from [5].

The model-based systems engineering (MBSE) approach takes it one step further. The V-Model does represent the general sequence of development, however, it is a simplified version lacking an important characteristic. It does not show the dependencies and relations of how the single elements on the same level of the V-Model interact with each other, which they do quite a lot. For example replacing a star tracker with a Sun sensor in a fictitious earth observation satellite project poses not only changes to the AOCS but to many other subsystems as well, as depicted in Tab. 1.1. One main motivation of MBSE is that having multiple models, e.g. mechanical, electrical, thermal and functional models, all existing in separate locations, complicates traceability of such design changes and requirements throughout all engineering domains. For a project with a certain complexity level, traceability of dependencies and interdependencies between components or higher level elements can be overly exhaustive and therefore error-prone, as relevant information is stored in extensive documents where it is hard to find. MBSE aims at combining all this information into one central data model where the relevant information for all domains is stored (see Fig. 1.2). Thus, as soon as one engineering domain changes a parameter of a component, the updated data is fed into the central data model and is visible for every stakeholder instead of being left unnoticed in an extensive document that is reluctantly accessed [4].

Another important aspect of MBSE is the close integration of tools to allow quick analysis of the system's behavior with respect to changes in the design. This does not only comprise functional behavior but also physical behavior. Being able to see the impact a design change poses on the overall system and its requirements ad hoc, allows to quickly compare different configurations and find the optimal one. This is another big advantage MBSE can have if implemented.

1.1 Motivation

In the recent past, such central data models where mostly created for single phases, mostly in the early development, using tools like Virtual Spacecraft Design (VSD), Open Concurrent Design Tool (OCDT), RangeDB, or Virtual Satellite [6]. However, these data models are usually abandoned after that development phase and the projects return to the document-centric approach for the further development process. This is mainly due to the fact that the tools are not flexible enough and not yet advanced enough to really support the whole development process [6].

Virtual Satellite (VirSat), being an MBSE tool developed by DLR-Institute of Simulation and Software Technology (DLR-SC), is aiming to be usable throughout the whole project lifecycle. For this, DLR-SC added the functionality to create own so called concepts inside the software. These concepts are defined to model different aspects of a system that are not included by default. Concepts implemented by default are for example mass budget, power budget, requirements or functional electrical architecture. By allowing the user to create own concepts, the software becomes more versatile and customizable for more detailed system models [6].

To integrate analysis tools as much as possible with the system model, a key purpose of Virtual Satellite is to provide access to the relevant system model data for engineering processes and tools to be executed, as depicted in Fig. 1.2. It is achieved by offering flexible interfaces for tools to connect to. Extending this integration of analysis tools, using the customizable interfaces VirSat provides is the main goal of this work.

1.2 Scope

To enable the usage of Virtual Satellite as an MBSE tool in all design development phases, advanced modeling capabilities and functionality have to be added to the software. This especially includes interfaces for other tools to access the conceptual data model (CDM), the central data model of Virtual Satellite.

The scope of this work is to enhance the capabilities of the MBSE approach in Virtual Satellite by developing the necessary infrastructure to include thermal models in the system model. Furthermore, it is part of the thesis to elaborate a concept to enable the execution of thermal simulations with CalculiX using data extracted from the Virtual Satellite CDM.

Testing and validating the newly established workflow and the linked tools and concepts is most beneficial for the outcome of this work. Thus, an example model shall be created using the newly implemented thermal modeling capabilities and the already present geometric modeling capabilities of VirSat

Therewith, this work provides several outputs. Firstly, the framework necessary for thermal engineers to use Virtual Satellite in future projects for modeling the thermal properties of a spacecraft. Secondly, a workflow concept is established to perform thermal analysis with the least possible amount of effort and least possible delay utilizing the system model from Virtual Satellite. And thirdly, as a side effect it is proved that Virtual Satellite can be tailored to individual project modeling needs, not only by software developers but also by project engineers with little or even no programming experience.

1.3 Kickoff Concept

For achieving the above described goals, four main tasks have to be fulfilled:

1. Virtual Satellite has to be extended in that way, that a thermal model can be stored in the CDM
2. The thermal properties and other relevant data for setting up a simulation have to be extracted from the CDM
3. A suitable input script file for the solver obeying the syntax of the simulation tool has to be generated
4. The simulation results have to be fed back to Virtual Satellite

As mentioned before, Virtual Satellite offers the ability to add custom concepts to the data model. These concepts can be defined to contain parameters, equations, or functional dependencies for example. They can be created using a rather top level programming language inside the Eclipse integrated development environment (IDE) of Virtual Satellite. Afterwards they can be included in the actual concept library among all other concepts.

The concept to be created for thermal analysis contains all important parameters for setting up the simulation. A list of these is provided in Tab. 1.2. Apart from the thermal parameters there is another important aspect that cannot be modeled purely by a parameter assigned to an element. The thermal interfaces between different components are not a material or element parameter but a combination of the contact pressure, the surface quality and the materials involved. Together, they can be expressed as the thermal contact conductivity. Knowing these conductivities is elementary for modeling the system correctly. To include this into the system model another feature is added to the concept, the thermal port. Each element in the system model can have one ore more thermal ports and each significant contact with another component is modeled by a connection between two ports. This connection is characterized by the thermal contact conductivity. This initial idea of a concept is depicted in Fig. 1.3. To complete the required data for a simulation, a top level class is created acting as a container for general simulation relevant information like static/transient simulation and specific post processing data to be obtained from the output.

Now, to make the thermal model usable for any simulation tool it has to be in the right form. For CalculiX that means model and simulation setup have to be combined in a script file. This script file has a certain syntax to make it readable for the solver. The concept for acquiring this is to write an application for Virtual Satellite that gets all necessary data, except for the geometry, and

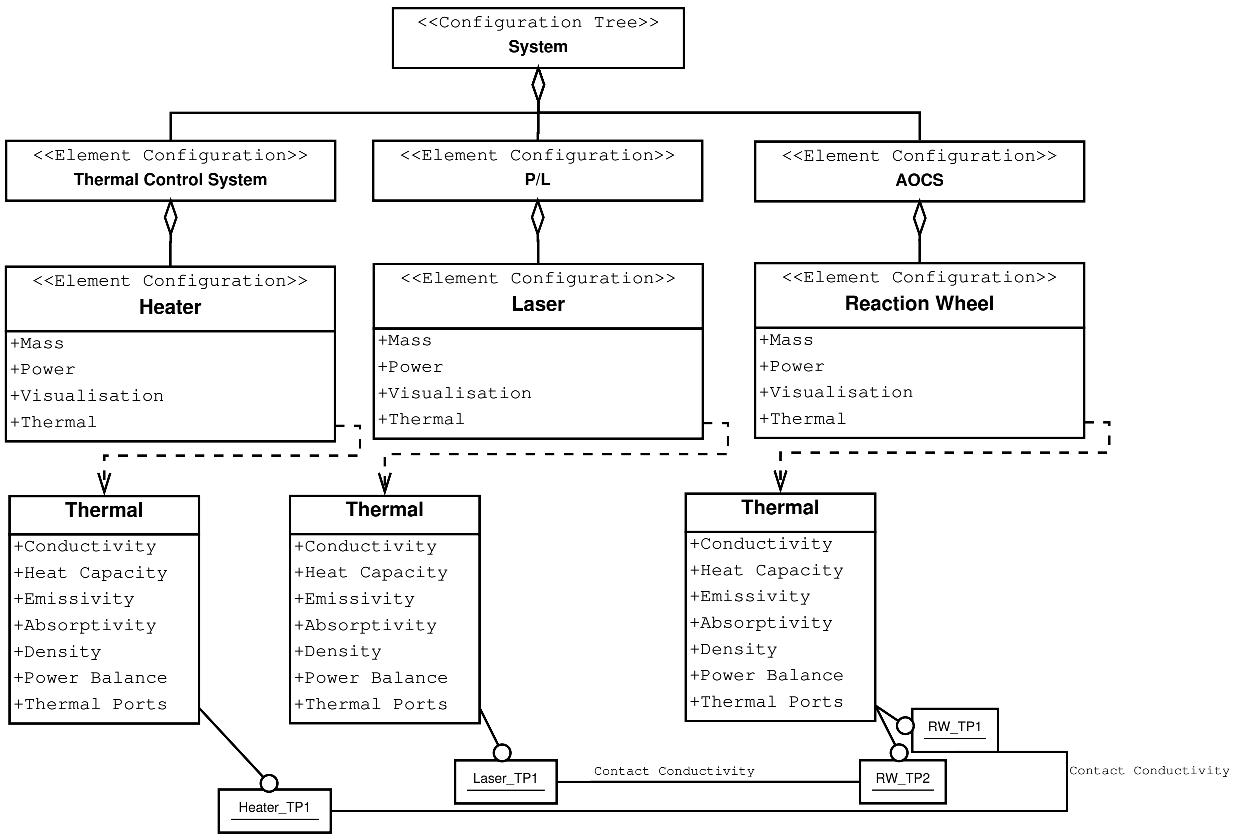


Figure 1.3: Draft for thermal concept, shown among some other concepts in a Virtual Satellite system model. Source: Own representation.

Table 1.2: Thermal parameters to be included in the concept

Symbol	Parameter	Unit
α	Absorptivity	-
c_p	Heat capacity at constant pressure	$\mathrm{W\,kg^{-1}\,K}$
ϵ	Emissivity	-
h_c	Thermal contact conductivity	$\mathrm{W\,m^{-2}\,K}$
λ	Thermal conductivity	$\mathrm{W\,m^{-1}\,K}$
P	Thermal power	W
ρ	Density	$\mathrm{kg\,m^{-3}}$
T_{start}	Initial temperature	K

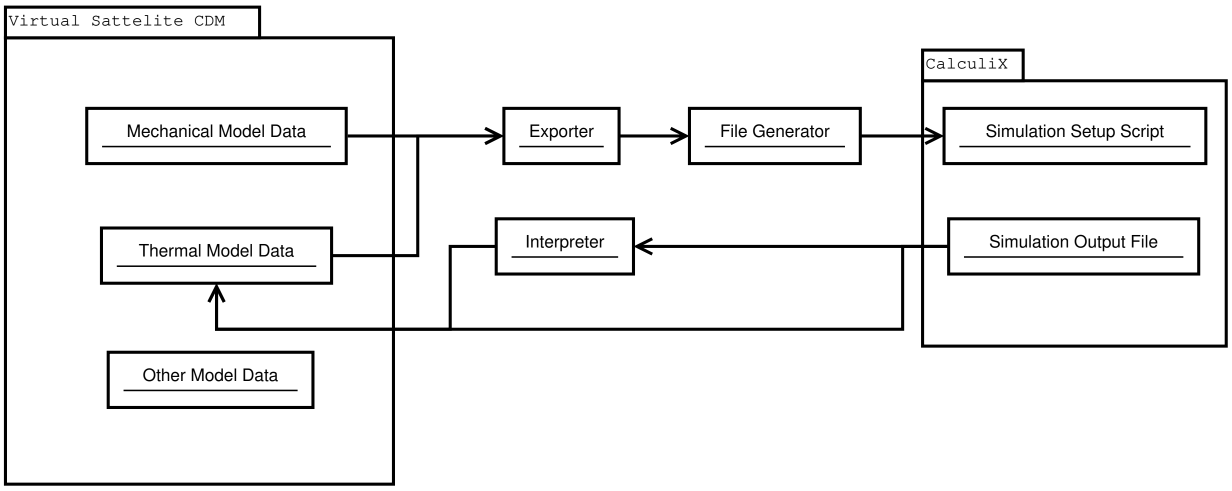


Figure 1.4: Overview of information exchange between VirSat and CalculiX. Source: Own representation.

writes it into a file in a specific order that matches the syntax of CalculiX. The geometry data is obtained through a different channel. For that, the visualization concept in Virtual Satellite and its export function can be used. This concept enables the user to model or import the geometries of all components in Virtual Satellite. The export function allows to export this geometry and alignment data along with the hierarchy data of the configuration tree. From this file the necessary information to build a mesh can be obtained. After meshing, the geometry file and the rest of the simulation setup file are merged to form the complete input file for CalculiX.

After successful execution CalculiX creates an output text file with all simulation results. This file is stored in the project repository to make it accessible in Virtual Satellite. Additionally, some kind of output interpretation would be beneficial for the readability of the simulation results. The exact function of this interpreter is not yet clear and has to be defined in the course of the thesis. A comprehensive overview of the information flow is depicted in Fig. 1.4.

1.4 Compasso Project

This thesis is written in the course of the Compasso project. Compasso is a payload to qualify optical technologies aboard the International Space Station (ISS) for future global navigation satellite system (GNSS) applications. The main components are an optical atomic iodine clock as well as a laser communication and ranging terminal for ground link [7]. The optical clock for example inherits the potential to be used in future GNSS programs due to the higher frequency stability, compared to the microwave clocks currently used in GNSS systems [8]. DLR-Galileo Competence Center (DLR-GK) develops MBSE in the course of Compasso to make use of the developed MBSE functionality in the future.

Chapter 2

MBSE Framework

This chapter introduces the whole framework of MBSE related to this work, starting with some general aspects about the methodology itself and the connection between MBSE and thermal analyses. In the second part of the chapter, Virtual Satellite, the MBSE tool developed by DLR-SC, is presented and its functions, especially those used in the further course of this work, are described.

2.1 Model Based Systems Engineering

One definition of model-based systems engineering is ”the formalized application of modeling to support system requirements, design, analysis, verification and validation beginning in the conceptual design phase and continuing throughout development and later life cycle phases” [4]. Its objective is to describe systems and different perspectives on the system (e.g. components, functions, modes, interfaces) with models, as depicted in Fig. 2.1, not with documents. Hence, the ultimate goal of MBSE is to replace document-centric systems engineering, which is the traditional approach where all data describing the system is located in different documents. By storing data in models that, when combined, represent as much of the system's properties as possible it is easiest to access, read and understand the different aspects of a system. Furthermore, the traceability of requirements and impacts of design changes is substantially enhanced. MBSE is expected to replace the document-centric approach in the future due to the increasing complexity of technical systems [9, 10].

A model of a system is a representation of it, that is intended to improve its comprehensibility [12]. Therefore, it must consider all parameters that are of importance for the application it was created for but can eliminate irrelevant information to help understanding a system faster. This definition rather wide and everything from a simple block diagram drawing to a highly advanced physical implementation of a system to conduct tests with is referred to as a model. In an MBSE context the term ”model” is mostly referring to graphical, mathematical or physical models describing the

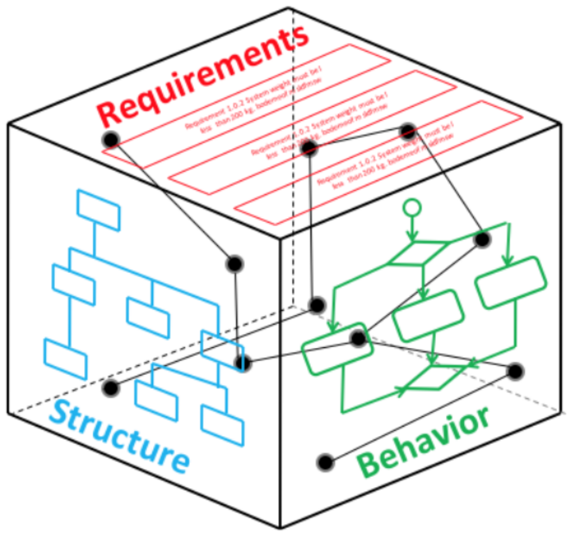


Figure 2.1: Visual representation of the system model as a multidimensional object consisting of different models. Source: Adopted from [11].

structure and behavior of the system in certain regards [13].

One important aspect of MBSE is that there is one central data model, which stores a number of single models in one central repository [13]. It is where all models are connected to each other. This repository always contains valid data since every change that is applied to one model is instantly forwarded to all other relevant models, maintaining a consistent set of data. Furthermore, calculated indicators of the system, like budgets, are recalculated to view the impact of a change in the model immediately. Applications for models in MBSE are widely spread and comprise architectural aspects like functions, behaviors, structure, interfaces as well as analytical aspects like cost or reliability [13]. Generally, it is always desirable to cover as many aspects of a system as possible with models since they provide some benefits compared to documents as discussed in the following section.

2.1.1 MBSE Benefits and Comparison with Document-Centric Approach

Compared to traditional document-centric systems engineering, MBSE has a number of benefits, which is why it is expected to eventually supersede the traditional methodology [9]. During system concept definition in the earliest phases, many stakeholders are involved in a project [13]. As some of those stakeholders might not have the appropriate professional background to interpret technical documents well enough for understanding how the system is intended to work, especially graphical models can deliver an easy to understand overview over a certain architectural aspect of a system. This eases work for all stakeholders since it is both easier to communicate how the system works with respect to their own domain, and also it is easier to understand the system's characteristics of other domains as well. In concurrent engineering studies this is a key advantage since there are experts of different fields having to understand the system in limited time. Thus,

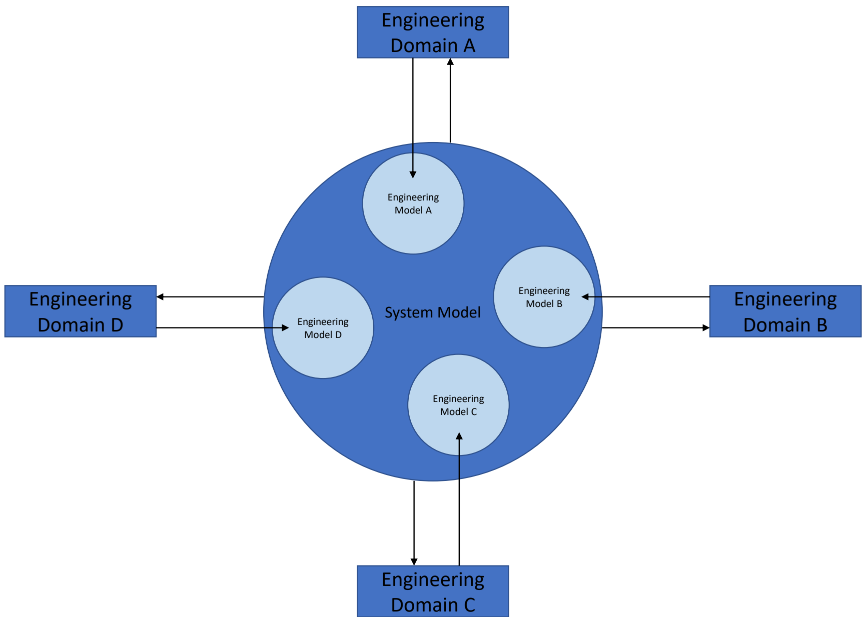


Figure 2.2: Visualization of the system model consisting of different domain models. All domains can access the whole model but only write to their own domain model. Source: Own representation

communication between different disciplines is eased significantly

One main intention of MBSE is enabling the possibility to track requirements and constraints from the main system model down to component level using mainly functional models and then check their fulfillment with analytical models like simulations or calculations. This traceability is a significant benefit over document-centric systems engineering as it is faster to obtain a complete overview over the fulfillment of all requirements on the one hand and it is also assured that all requirements are actually respected on the other hand, since the relevant property a requirement refers to is directly linked to it in the model.

As Fig. 2.2 indicates, another advantage of MBSE is that data stored in the central system model can be readily accessed by all engineering domains and directly used to simulate the system's behavior in their domains without having to access different documents to obtain this data. Conceivable applications for this would be state models, electrical models, mechanical models or thermal models to name just a few. Some of them were already implemented in MBSE tools like VirSat. The goal of MBSE is to conduct all analyses and as many verifications as possible (e.g. by means of simulations) with the one system model as data source, to completely turn away from the document-centric approach.

The ideal case is to execute all analyses and simulations automatically as soon as a change in the design is committed to the system model, so that all consequences of that change, on the requirements for example, can be monitored as fast as possible. This benefits especially systems engineers, as every system analysis that is conducted automatically saves time, which can then be spent on other tasks. But also subsystem engineers profit. The more simulation results provided in the early phases without additional required man-hours, the better the knowledge of the system and the better the requirements of each subsystem can be determined. As a consequence, substantial changes in the design can be expected to be less likely, which in turn means a more efficient early design phase. This is also what this work aims at achieving.

By achieving the possibility to run certain analyses early in the development phase, design flaws can be detected earlier than without using MBSE. Since the cost of errors rise with the project progress, this can also save significant amounts of resources [14].

In the later design phases MBSE could help setting up more sophisticated analytical simulations, automating certain tasks and reducing the work necessary to finish the setup. This work can be regarded as an example on how this could be applied.

2.1.2 Current State of MBSE

The transition from document-centric systems engineering to MBSE is currently in progress. Major companies, Ford Motor Company as an example, are adapting the methodology to tackle the challenge of handling complex systems [9]. Other companies certainly use it to some extent, without having it explicitly named as such. The most dominant tool for creating models for systems engineering is the SysML language [15]. With this, it is possible to express relationships between objects enabling the engineer to display system architecture and process flow and trace down requirements [15]. International Council on Systems Engineering (INCOSE) compares the current state, as of 2014, with the early phase of the evolution of computer aided engineering (CAE)/ computer aided design (CAD) [9] and Systems Engineering Body of Knowledge (SEBoK) states that MBSE is ”applied in pockets within organizations and unevenly across industry sectors.” [10]. SEBoK furthermore states that there is still some room for improvement to be covered for MBSE tools and methods to widely establish the methodology throughout all industries [10]. Indeed there are many new ideas for new ways of applying MBSE to space project development and effort is spent to improve the capabilities of current MBSE tools like VirSat. New releases are published continuously, introducing new modeling capabilities as well as new interfaces to other modeling tools. But not only the core developer team is working on improving VirSat, also other Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR) institutes undertake efforts to add functionality to it. One example for this are the initiative to add a database of spacecraft components to VirSat that can be easily selected, automatically adding the data from their datasheet to the system model [16]. Another example is this work, as it aims to add additional content to VirSat as well.

2.1.3 Thermal Analysis and MBSE

MBSE referring mainly to classical systems engineering activities so far, technically does not explicitly cover thermal analyses since they are part of the tasks of thermal engineers. However, thermal analyses in the context of this work are applied to early system models in the first place. Furthermore, in MBSE basically everything is linked to the central data model and therefore to systems engineering. As the thermal analyses in this work are first considered more as relatively quick analyses to see the expected magnitude of thermal parameters and the impact of design changes, they are moved much farther into the systems engineering domain and are essentially located on the border between the two domains. However, in the end, among other factors, the amount of detail of the system model determines the accuracy of the analysis, as discussed in chapter 5.3. Hence, both systems engineering and classical thermal engineering domains are part of the presented workflow. Nevertheless, the effort both have to spend on the generation of such a simulation is ought to be low and the presented workflow aims to reduce it as much as possible. As a result of the points raised, this work aims to create a workflow applicable as early as for phase A studies, but also for the following early design phases as well as building a foundation for later design phase analyses. Therefore, the workflow is designed to be as flexible as possible to allow the inclusion of further aspects of a thermal model in the future to cover as much of the project lifecycle as possible.

2.2 Thermal Analysis

In space a harsh environment is prevalent in many respects. Radiation, energetic particles, vacuum and extreme temperatures, the latter one being the primary concern in this work, impose high demands on a space system's component survivability [17]. Temperature in space is often subject major variations [18]. Especially in certain short periodic planetary orbits, like low Earth orbits with low inclinations, the persistent switch between solar eclipse and solar presence is problematic for most components since they are designed to work in specific temperature ranges [19, 20]. Fig. 2.3 gives an overview of the thermal environment a spacecraft encounters in space, with solar radiation, Earth albedo, and Earth infrared radiation.

Due to the absence of atmosphere around the spacecraft there is no reservoir apart from the space system's mass that can act as a heat buffer, keeping the temperature variations in limits. Neither is there matter that absorbs some fractions of the solar radiation before it reaches the spacecraft. Additionally, convection is completely absent in space, since no medium for it is present. The energy input during daytime is then cycled with essentially zero energy input from the Sun in eclipse, causing high temperature gradients on the outer spacecraft components. As heat conduction being present inside a space vehicle, the inner parts will be loaded by the temperature gradients acting on the outer walls as well.

Another challenge for the spacecraft is that since there is no convection and no conduction outside the spacecraft, the only way to efficiently lose heat is radiation.

In the following, the need for temperature control and thermal analysis, as well as their functioning is discussed. Afterwards, the simulation software used for thermal simulations in this work will be introduced. First however, the main principles of thermodynamics that are relevant in this work are briefly presented.

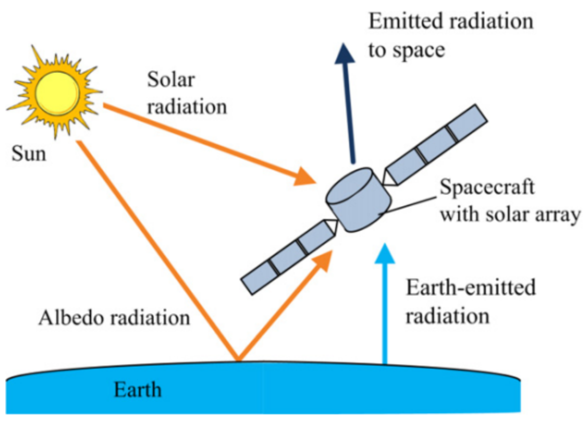


Figure 2.3: A spacecraft's thermal environment with the radiative heat flows acting on the surface of the spacecraft. Source: Adopted from [21].

2.2.1 Thermodynamic Foundations

To understand the thermal relationships within a spacecraft, one has to study the thermodynamic processes happening aboard, of which conduction and thermal radiation are the most important ones. They govern the temperature distribution throughout the spacecraft in most cases [18]. This section covers all basics necessary for understanding the thermal aspects of a space system.

2.2.1.1 Temperature and Internal Energy

Temperature is a state quantity and refers to the amount of motion the atoms or molecules of an ideal gas experience [22]. The higher the temperature, the higher the mean motion of the particles. It is important to note the term motion means the average random movement in all directions, like vibration or rotation, opposed to a velocity, which is a directional movement in one specific direction [23, 22]. Thus, a fast-moving particle does not necessarily have a high temperature.

According to the first law of thermodynamics, the change in internal energy, which is the state variable giving the amount of energy that is available for a thermodynamic process, is formed by the amount of work done on the system and the amount of heat transferred to or from the system [24]. As the work done relates to the volume change, which is assumed to be negligible for solid bodies, compared to gases, heat transfer is the only effective process to change the amount of internal energy. According to 2.1 the transferred heat and thus the change in internal energy dU is composed of the temperature T and the change in entropy dS as scaling factor [24]. Here, p is the pressure and dV the volume change. Consequently, temperature is the main indicator for the amount of internal energy that is present in a system. And since temperature is an intensive value, it serves as an indicator for the local distribution of the energy throughout the system, making it a desirable characteristic for examination [23].

Table 2.1: Examples for the thermal conductivities of different species [25]. Note that the values for metals vary significantly depending on the alloy [17].

Material	Thermal conductivity $\mathrm{W\,m^{-1}\,K^{-1}}$
Diamond	1000.0
Copper	385.0
Aluminum	205.0
Steel	50.2
Fiberglass	0.04
Air at 0 °C	0.024
Polyurethane	0.02

$$dU = TdS - pdV \tag{2.1}$$

2.2.1.2 Thermal Conduction

As stated before, temperature represents the amount of vibration and rotation a particle experiences. If particles of high temperature and low temperature are close to each other, it is likely that the intensively vibrating high temperature particle collides with the low temperature particle. When this happens some impulse of the hot particle is transferred to the cold particle, inciting it to vibrate as well, resulting in a raise of its temperature [22]. The former hot particle loses some of its energy and is now colder than before. This is the process of heat conduction [18].

Now, the engineering approach to this phenomenon is more macroscopic. It is applied in the form of thermal conductivity, which is a material parameter for how much heat is transferred over a certain distance, given a certain temperature difference and a cross section area [23]. Its value is influenced by different material properties, the most important being the molecular bonding and the amount of free electrons [18]. This already implies, that metals, generally having a high amount of free electrons are good thermal conductors, which is true [17]. Diamond, being among the most effective thermal conductors, has an exceptionally high thermal conductivity of around three times the value of copper, which is among the best metallic conductors [18, 25]. The difference in thermal conductivity among common materials, even if they are not intended for thermal application, is very high and ranges from far less than 1 $\mathrm{W\,m^{-1}\,K^{-1}}$ to more than 400 $\mathrm{W\,m^{-1}\,K^{-1}}$ as listed in Tab. 2.1 [25].

Macroscopically expressed, the heat flow $\dot{Q}$ between two points 1 and 2, according to Fourier's law in 2.2, can be expressed in terms of the temperature difference ΔT between them, the thermal conductivity λ, the cross section area of the material between the points A and the distance between the points Δx [18].

$$\dot{Q} = \frac{\lambda \cdot \Delta T \cdot A}{\Delta x} \tag{2.2}$$

In the general three-dimensional case, conduction is described by the differential equation in 2.3 where q_v is the heat flow per unit area, ρ the density, c_p the specific heat capacity and λ the conductivity of the material in the specific direction [18].

$$q_v = \rho \cdot c_p \cdot \frac{\partial T}{\partial t} + \lambda_x \cdot \frac{\partial^2 T}{\partial x} + \lambda_y \cdot \frac{\partial^2 T}{\partial y} + \lambda_z \cdot \frac{\partial^2 T}{\partial z} \tag{2.3}$$

Equation 2.4 shows how 2.3 can be expressed as a matrix equation.

$$Q = C_T \dot{T} + K_T T \tag{2.4}$$

Here Q is the matrix of heat flows, C_T represents the heat capacities, and K_T is the conductivity matrix.

2.2.1.3 Thermal Contact Conduction

The mechanisms of thermal conduction only work within closed bodies. As soon as there is a gap, even if it is very small, conduction will be inhibited, as the distances that can be covered by particles are too small to bridge the gap [18]. However, if two separate bodies are pressed against each other, they do have some contacts close enough for conduction to take place as depicted in Fig. 2.4 [18]. The magnitude of this contact conductivity depends on a whole variety of factors including pressure, surface quality and interstitial materials between the contacts [18]. There are some models for calculating the thermal contact conductivity, however they are all quite sophisticated and only valid for certain conditions and materials [17, 18, 26]. Due to the lack of universal models for the value of the thermal contact conductivity, is either determined experimentally or the thermal engineer uses similarity to already examined contact pairs for each individual contact [17].

2.2.1.4 Thermal Radiation

If a particle has a temperature above 0 K there is a certain chance that it spontaneously emits a photon [27]. The wavelength of the photon and therefore its energy primarily depends on the temperature of the particle [27]. The emission of photons fed with internal energy is called radiation. Max Planck discovered the formula for the emissive power of a blackbody at a certain temperature for a certain wavelength. The spectrum of the emissive power for the Sun is displayed in Fig. 2.5, that shows a typical shape for a Planck curve [28]. A blackbody is an idealization of reality which perfectly emits and absorbs radiation in all wavelengths, so that its radiative spectrum is

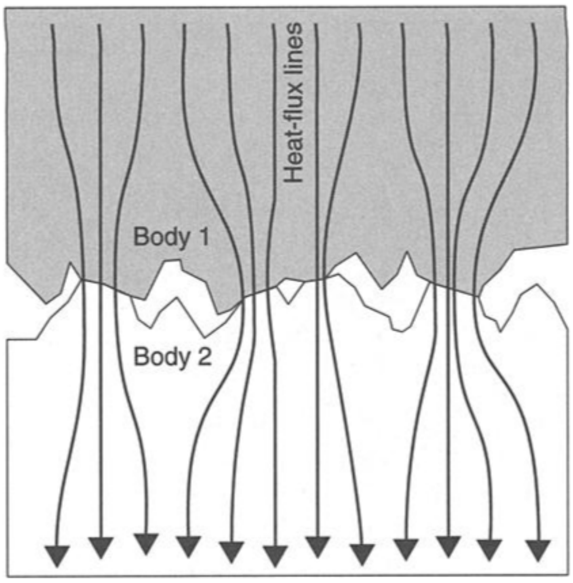


Figure 2.4: View on the cross section of a thermal contact. Heat flow by conduction is only enabled in the area of contact. Source: Adopted from [18].

purely defined by Planck's law [24]. This relation is shown in 2.5, where *I* is the spectral radiance, *c* is the speed of light, *h* is the Planck constant, *k* is the Boltzmann constant, λ is the wavelength and *T* is the temperature of the body [29].

$$I(\lambda, T) = \frac{2hc^2}{\lambda^5} \frac{1}{e^{\frac{hc}{\lambda kT}} - 1} \tag{2.5}$$

The wavelength spectrum of the emitted photons is generally lower, when the temperature of the particle is higher. This is covered in Wien's displacement law, which gives the relation between the maximum of the spectrum of emitted photons (λ_{max}) and the temperature for a blackbody (*T*) as it is depicted in 2.6 [18]. Wien's displacement law is actually directly related to Planck's law as it determines the maximum point of Planck's curve.

$$\lambda_{max} = \frac{2897.8\,\mu\mathrm{m\,K}}{T} \tag{2.6}$$

Ideal blackbodies are a theoretical construct and real objects can only be approximated as blackbody citeSCThermalControl. The Sun for example has an effective surface temperature of around 5800 K [30]. The calculated blackbody temperature according to Wien's law(2.6), rearranged for temperature 2.7, assuming a maximum of solar radiation at roughly 0.5 µm calculates to 5795.6 K, which is close to the real temperature [30].

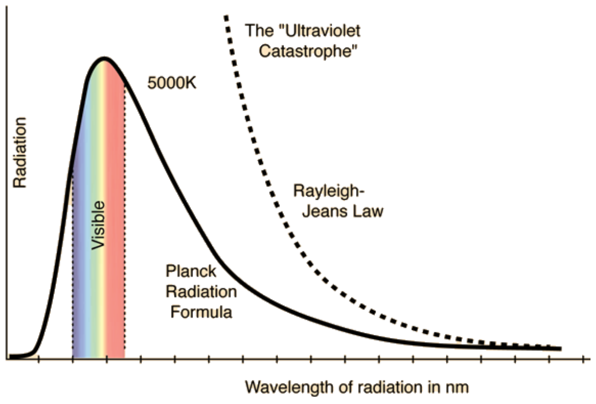


Figure 2.5: Planck curve displaying the radiation energy emitted in a specific wavelength regime. Source: Adopted from [28]

$$
\begin{aligned}
T &= \frac{2897.8\,\mu\text{m}\,\text{K}}{\lambda_{max}} \\
&= \frac{2897.8\,\mu\text{m}\,\text{K}}{0.5\,\mu\text{m}} \\
&= 5795.6\,\text{K}
\end{aligned}
\tag{2.7}
$$

Non-ideal blackbody surfaces are approximated as a so called gray surfaces [24]. The non-ideal radiative properties are expressed as emissivity and absorptivity factors ranging between zero and one that are multiplied with the ideal blackbody radiation to account for the lower absorptivity and emissivity [24]. The Stefan-Boltzmann law is the integral of Planck's law over all wavelengths and gives the total emitted power of a surface area for a certain temperature. It is denoted as shown in 2.8 where A is the radiating area, ϵ is the emissivity factor of a gray body, P is the emitted power, σ the Stefan-Boltzmann constant and T is the actual temperature of the radiating surface [18].

$$
P = \frac{A \cdot e \cdot T^4}{\sigma} \tag{2.8}
$$

On earth, conduction and convection are dominant over radiation in most cases. In space however, radiation is the main mechanism for a spacecraft to lose energy since conduction and convection cannot be used to dump energy to the environment.

2.2.1.5 Specific Heat Capacity

If one wants to analyze the temporal behavior of a system in terms of thermal aspects, there is one important material parameter that must be included in the equations. This is the specific heat capacity. It is a measure for the amount of energy a material can absorb to experience a certain increase in temperature and expressed in SI-units it is $\mathrm{J\,kg^{-1}\,K^{-1}}$ [23]. Consequently, as energy equals power multiplied by time, it influences the temporal behavior of the system. Generally, the heat capacity is not constant with respect to the temperature [23]. These variations are in most temperature ranges relatively small, however in the regimes of evaporation/condensation and freezing/dewing the specific heat capacity can vary strongly with only small temperature changes when a phase change is involved [23]. In addition, compressible agents like gases have different heat capacities for heating with constant volume (c_v) or heating with constant pressure (c_p). As spacecraft components are usually in their solid state, and kept around a relatively small temperature variation, usually the c_p value is used for analyses [17].

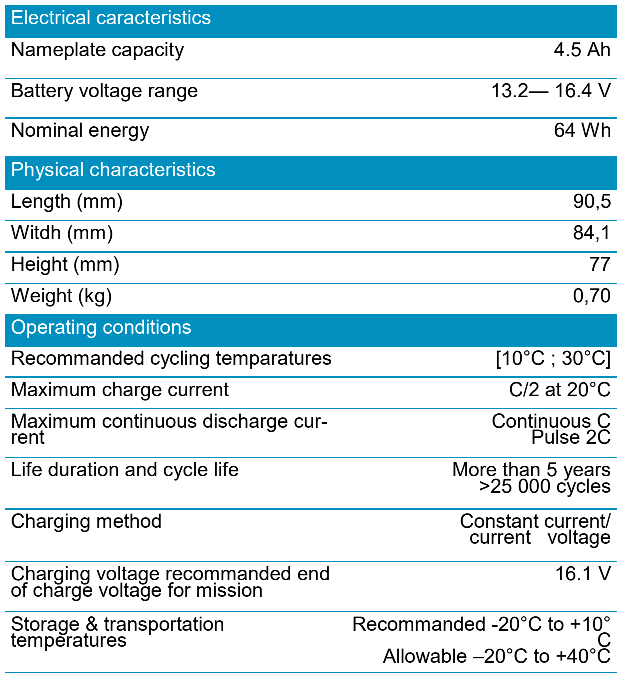
2.2.2 Thermal Analysis and Thermal Control in Space Systems

Many components and experiments need an exceptionally stable environment to work properly. This holds for kinematic aspects, as well as for magnetic or thermal aspects [31]. In addition, with the miniaturization of integrated circuits, thermal issues evolve for electrical components as the power converted to heat is applied to a smaller area. This already indicates the need for thermal analysis. The thermal control system of a component should not dump too much heat nor too little. It should not take too much space and it must be compatible with other components of the system.

All components have specified operating temperatures, which apply to mechanical as well as to electrical, electronic and electromechanical (EEE) components [17]. But it is especially important for electrical components, since their operating temperature range is usually relatively narrow and their lifetime is substantially decreased when the operating temperature is exceeded. Commercially available batteries for space applications for example have a recommended range of operating temperature between 10 °C and 30 °C as it is visible in Fig. 2.6 [19].

Some space components utilizing ultra stable lasers for example, such as high precision atomic clocks or laser-cooled experiments, are particularly sensitive to temperature changes resulting in temperature requirements in the sub Kelvin regime [32, 33]. For a hydrogen maser clock, the temperature needs to be stabilized within 50 mK for a time period of more than one day, while the master oscillator laser used to cool atoms and perform interferometry, among other functions, in the MAIUS-1 project had to be maintained within a range of 0.2 K [17, 34]. These components usually have dedicated thermal control subsystems (TCSs) that are somewhat independent of the general spacecraft TCS in that sense, that the temperature requirement on the whole clock unit is only in the range of some ten Kelvin instead of less than one [17]. An example for such a component with dedicated temperature control is shown in Fig. 2.7, where the PHARAO atomic clock's temperature ranges of the subcomponents are visible.

In space, the general absence of convection and conduction to dump heat from the system makes it even harder to design an appropriate TCS, especially keeping the temperature relatively stable



Electrical caracteristics	
Nameplate capacity	4.5 Ah
Battery voltage range	13.2— 16.4 V
Nominal energy	64 Wh
Physical characteristics	
Length (mm)	90,5
Witdh (mm)	84,1
Height (mm)	77
Weight (kg)	0,70
Operating conditions	
Recommanded cycling temparatures	[10°C ; 30°C]
Maximum charge current	C/2 at 20°C
Maximum continuous discharge current	Continuous C Pulse 2C
Life duration and cycle life	More than 5 years >25 000 cycles
Charging method	Constant current/ current voltage
Charging voltage recommanded end of charge voltage for mission	16.1 V
Storage & transportation temperatures	Recommanded -20°C to +10°C Allowable –20°C to +40°C

Figure 2.6: Extract from the data sheet of a battery for space use. Source: Adopted from [19]

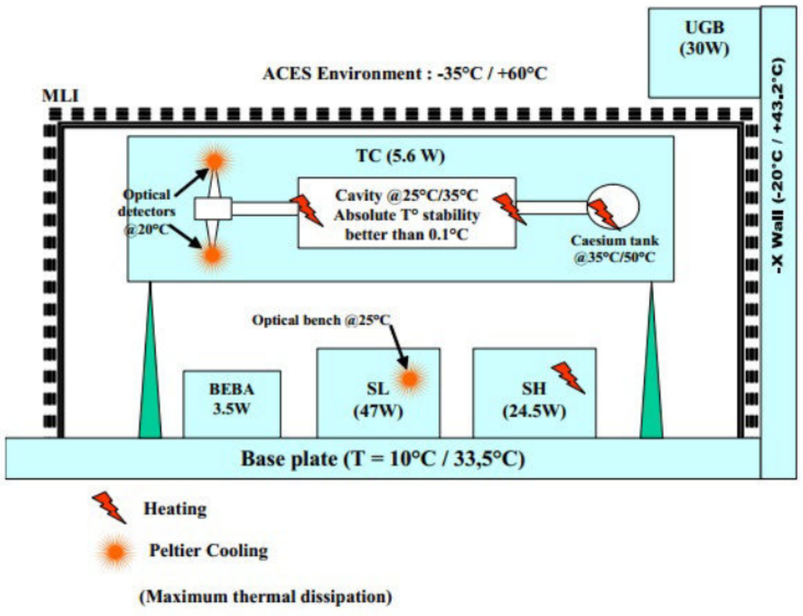

UGB (30W)
ACES Environment : -35°C / +60°C
MLI
-X Wall (-20°C / +43.2°C)
TC (5.6 W)
Optical detectors @20°C
Cavity @25°C/35°C Absolute T° stability better than 0.1°C
Caesium tank @35°C/50°C
Optical bench @25°C
BEBA 3.5W
SL (47W)
SH (24.5W)
Base plate (T = 10°C / 33,5°C)
Heating
Peltier Cooling
(Maximum thermal dissipation)

Figure 2.7: Layout of the internal TCS of the PHARAO atomic clock. Source: Adopted from [32].

is a challenge, since there is no atmosphere balancing variations. Thus, numerous sophisticated components for temperature control were developed for space applications [35].

To meet the requirements of all components, especially considering that different components may favor different operating temperatures, it is not sufficient to simply estimate the amount of cooling or heating needed. Thus, mathematical models must be created, representing the conditions the system experiences in space.

2.2.3 Numerical Methods

There are different ways of tackling the issue of calculating the temperatures to be expected on a spacecraft. It is certainly possible to calculate simple systems with not too many boundary conditions analytically without help of a computer. However, with increasing spacecraft complexity this approach quickly becomes impractical. Computers, being predestined to solve large systems of equations are the method of choice for executing thermal analyses. Two methods for solving such problems are presented in the following, the finite-differences method (FDM) and the finite element methods (FEM). However, the FDM will be only covered briefly, since the FEM was predetermined for this work as it is a multi-purpose method that is readily available in open source software packages.

2.2.3.1 Finite Differences Method

The FDM is a method to solve systems of differential equations numerically. The underlying approach of finite differences is to replace all first order derivatives by the difference expression denoted in 2.9 and all second order derivatives by the difference expression denoted in 2.10 [36].

$$f'(x) \approx \frac{f(x+dx) - f(x)}{dx} \tag{2.9}$$

$$f''(x) \approx \frac{f(x+dx) - 2f(x) + f(x-dx)}{dx^2} \tag{2.10}$$

By replacing all differential expressions in the differential equation with difference expressions it becomes easier to solve with a computer. The next step is to arrange the single terms in a way, that each of the discretized function values has its mathematical operations as one combined factor. For 2.10 this would look like denoted in 2.11.

$$\frac{df(x)}{d^2x} = \frac{1}{dx^2} f_{i-1} - \frac{2}{dx^2} f_i + \frac{1}{dx^2} f_{i+1} \tag{2.11}$$

With this kind of expression set up for all discrete points to be examined, a matrix equation system can be obtained, which can then be solved using the boundary conditions.

2.2.3.2 Finite Element Method

The most popular and advanced method for solving differential equation problems is the finite element method. The first literature was released in the 1960s and it is still used nowadays having a wide range of applications from structural integrity calculations to field problems like thermal analyses [37].

The idea is to split the real system with infinite degrees of freedom into a finite number of sub-elements with a finite number of degrees of freedom. The sub-elements are bounded by nodes, which are discrete points. The original differential equations can now be discretized by applying them to the set of discrete nodes. Having a set of equations in the form of 2.12 finite element method can be used to solve it approximately [38].

$$Kt = q \tag{2.12}$$

In this case K is a matrix of differential operators that is multiplied with t, the vector of degrees of freedom for each node, which is the temperature for a thermal simulation. The result of this multiplication is q, which is the known vector of external boundary conditions. To obtain a discrete system of equations, the variable or variables representing the desired continuous solution function are replaced by N sets of variables, representing the desired solution for N discrete nodes. Where the number N is the actual number of nodes that were created with the mesh [38].

To cover the area in between these nodes, the continuous basis function $\phi(x, y, z)$ is introduced and multiplied with the solution variable [38]. It is a continuous function that can be linear or of higher order. The basis function equals one on the coordinate of the actual node it belongs to and 0 on the coordinates of all other nodes [38]. That means for every point within the structure, the approximate solution is calculated by the sum of the values at the N different nodes multiplied with their respective basis functions. The value of the latter varies depending on their order and the coordinates. For a linear basis function the value decreases linearly to zero between two nodes. After discretization the function looks like in 2.13 [38].

$$K \begin{bmatrix} t_1 \cdot \phi_1(x, y, z) \\ t_2 \cdot \phi_2(x, y, z) \\ \vdots \\ t_N \cdot \phi_N(x, y, z) \end{bmatrix} \overset{!}{=} q \tag{2.13}$$

The exclamation mark added to the original equation already indicates that after discretizing, the left side is no longer exactly equal to the ride side. By discretizing the equation in most cases only an approximate solution can be obtained. This is one reason why finite elements never deliver the exact solution but always an approximation [38]. Rearranging 2.13 with respecting the fact that it is only an approximation introduces a residual R as denoted in 2.14.

$$K \begin{bmatrix} t_1 \cdot \phi_1(x,y,z) \\ t_2 \cdot \phi_2(x,y,z) \\ \vdots \\ t_N \cdot \phi_N(x,y,z) \end{bmatrix} - q = R \tag{2.14}$$

To obtain the best possible solution, the goal of finite element method is to reduce the residual, since the lower the residual, the better the fit of the approximated solution to the actual solution.

Coming from one differential equation the matrix representation actually only represents an equation system with one equation and N variables. To make up for this another function similar to the basis function is introduced, the weighing function. More specifically N weighing functions that are multiplied with the solution vector, just like the basis function [38]. With these weighing functions there are now N equations with N variables, making the system well-determined. Depending on the type of finite element method, the actual content of the weighing function differs. For the Galerkin method, which is a common method of solving the finite element equations, the weighing function is just the same as the basis function [38].

2.2.3.3 Finite Element Meshes

Meshes are the discretized representation of a real system and therefore the model of the geometry the finite element method is applied to [39]. As such, it is intended to be as similar to the original system as possible. The mesh consists of elements which again consist of nodes. Commonly used elements in three-dimensional analysis are tetrahedral and hexahedral elements comprising three or four nodes for each face of the element, respectively [39]. As the discretization error of the analysis rises the more deviations between mesh and real geometry exist, it is generally desirable to have a fine mesh. However with decreasing mesh size the number of elements increases exponentially which increases the size of the equation system and therefore the computing time for the analysis. Thus, meshing is always a trade-off process between model accuracy and computing effort.

To reduce the necessary computing effort but still obtain good solution, local mesh refinement can be utilized. With this, important areas in the model can be chosen for a local increase in mesh accuracy without inflating the mesh with less important elements. The benefit regarding relatively large components with comparatively small critical areas is visible in Fig. 2.8, where the third mesh has significantly less elements than the second, but the number of elements in the critical area is similar.

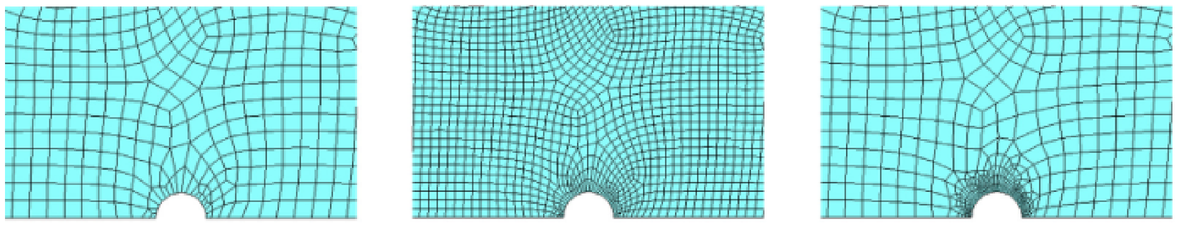


Figure 2.8: Comparison between three different meshes. First one is a coarse mesh, second one a globally refined mesh and the third one is a coarse mesh with locally refined areas. Source: Adopted from [40].

2.2.3.4 Mesh Convergence

Mesh convergence evaluates a qualitative aspect of the mesh [41, 42]. To determine if a mesh converges sufficiently, the behavior of the target value is examined with respect to the number of mesh elements. That means, multiple simulation runs are executed while increasing the number of elements in the mesh. If the deviation of the target value with respect to the previous simulation is under a certain criteria value, the mesh is accepted as convergent [41].

2.2.4 CalculiX

With the finite elements method existing for more than 50 years, there are numerous commercial and open source finite element pre-processors, solvers, post-processors and software packages. CalculiX is one of the open source representatives and was developed from the late 1990s on by Guido Dhondt and Klaus Wittig and is still updated regularly with the last update released in 2020 [43]. CalculiX is a package consisting of pre-processor, multiple solvers and a post-processor to solve linear and nonlinear mechanical, thermomechanical or thermal problems. However, it is also possible to utilize it for other problems like computational fluid dynamics (CFD) [43].

CalculiX essentially consists of two parts, CalculiX CrunchiX and CalculiX GraphiX. The latter one is used for pre- and post-processing of the analysis and offers a graphical user interface (GUI), while CrunchiX is the simulation tool without GUI. Since in this thesis pre-processing is done using VirSat and FreeCAD, the pre-processor is of no further interest. The post-processing capabilities however may be used for processing the results of the analysis.

To combine GraphiX with CrunchiX one can use the CalculiX launcher software, allowing to open an input file, executing the simulation and afterwards launching GraphiX to view the simulation results. Some more functions are available but most of them are designed for Linux and are not useable with a Windows operating system.

The mentioned input file is an alphanumeric file where all information necessary to execute the simulation is stored in such way, that it can be read by CalculiX CrunchiX. Its syntax is inspired by that of an Abaqus input file which is a commercial finite element analysis (FEA) software with similar areas of application as CalculiX. The exact structure of such an input file is presented in this chapter.

CalculiX's standard solver is the Spooles solver and is used in this work, however some additional solvers are supported as well and can be used after manual installation. [42].

All information in the following sections, that is needed for the syntax and structure of an input file is obtained from [42] and the example model provided with the CalculiX software package, if not explicitly marked as a different source.

2.2.4.1 Input Script

Many, if not all, finite element software packages offer a possibility of scripting the simulation setup [44, 45]. CalculiX has this capability of reading beforehand created input files as well. The scripting language used for this is inspired by the one that commercial FEA software Abaqus uses to conduct analyses. As discussed before, scripting a simulation setup can significantly reduce the time spent for setting up simulations in certain cases. However, it is somewhat time consuming to set up the script initially.

In CalculiX, the input file consists of three main sections. The first one is the model geometry. Then follows the thermal model data as well as general simulation configuration data. Appendix A is dedicated to these three parts of the input file and shows the structure of the input file for CalculiX in detail. The concrete sequence in which the simulation is set up is variable and many steps can be interchanged with one another. The sequence described in the appendix is how the simulation is set up for this work and is not the only possible sequence.

2.3 Virtual Satellite

To increase the efficiency of spacecraft development DLR-Institute of Simulation and Software Technology (DLR-SC) developed the MBSE tool VirSat. It was mainly aiming on supporting concurrent engineering studies, nevertheless, the intention to use the software for later development phases was already there from the beginning [46]. Several new functions were added to achieve this goal, supporting spacecraft development throughout all phases in various ways. Some of these functions will be presented in this section.

Fig. 2.9 depicts the phases in which VirSat is used compared to other MBSE tools that are more focused on the earlier project phases. The figure also shows that one intention of this work is to enable the execution of thermal simulations at an earlier stage of spacecraft design, as it provides the tools to perform analyses even without using domain specific design tools before.

Virtual Satellite (VirSat) 4 Core is a software offering a customizable data model for storing relevant engineering data of a satellite on the one hand and interfaces to access this data on the other hand. This strongly supports the idea of MBSE, as it provides one central data model where model data of different engineering disciplines can be found.

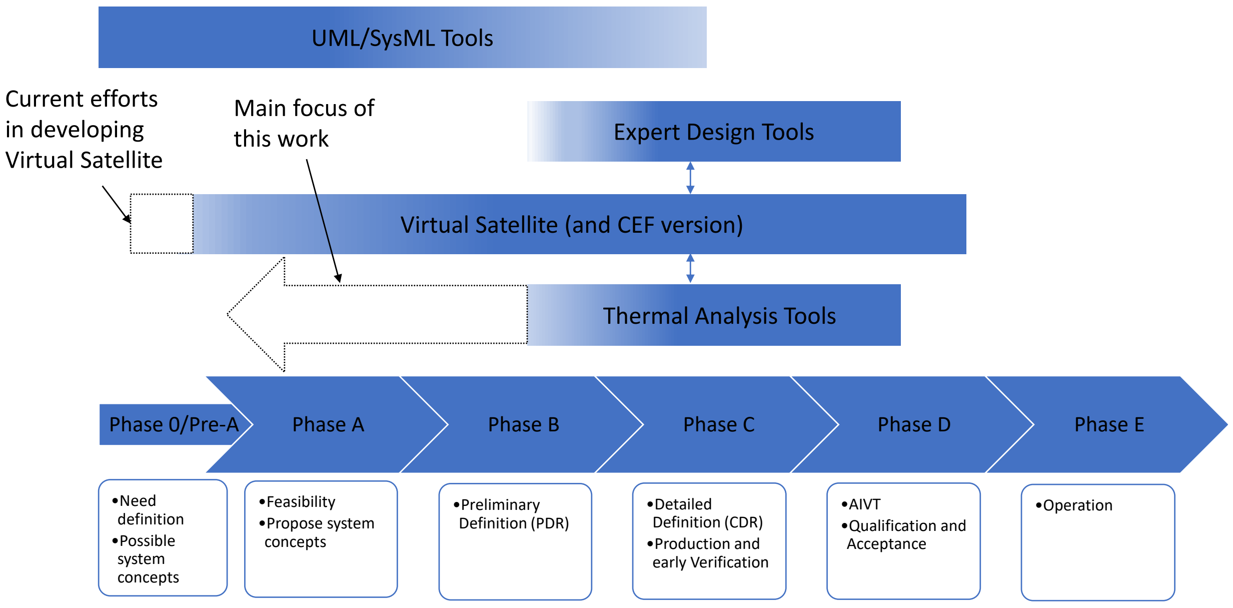

UML/SysML Tools
Current efforts in developing Virtual Satellite
Main focus of this work
Expert Design Tools
Virtual Satellite (and CEF version)
Thermal Analysis Tools
Phase 0/Pre-A
Phase A
Phase B
Phase C
Phase D
Phase E
•Need definition •Possible system concepts
•Feasibility •Propose system concepts
•Preliminary Definition (PDR)
•Detailed Definition (CDR) •Production and early Verification
•AIVT •Qualification and Acceptance
•Operation

Figure 2.9: Overview of the information flow in the workflow. Source: Own representation.

The framework or meta-model of this data model, the conceptual data model (CDM), is the core of VirSat. It is the blueprint for the system model, the actual implementation of the CDM. Within the CDM every aspect of the project to be stored in the system model is defined. This holds for the general hierarchical project structure as well as for individual properties of components. What defines VirSat besides the customizability of the CDM, allowing to add new aspects to tailor it to the individual project needs, is that it has a version control system (VCS) to enable the engineers to always restore an old system model state if a change does not have the desired effects [47].

The possibility to tailor the CDM makes VirSat a versatile tool. The ongoing effort of the developers to extend the concepts in VirSat and its customizability enabled it to be used not only in concurrent engineering but also for MBSE during later development phases as indicated by the "Product Structure" concept introducing project trees for different project phases. The concept is briefly presented, among the other concepts, in the following section.

One key aspect of VirSat is the role management. A project is represented by one of the "Product Structure" trees, where the subsystems can be assigned to certain engineering domains. Each user belonging to a domain can edit engineering data of their own domain only, but can read data of all other domains as well. This ensures, that the engineers always have an overview of the complete system, and data integrity is maintained, as only domain experts have the rights to perform changes within their domain.

Another important feature of VirSat are concepts. Concepts represent the individual engineering disciplines' relevant data to be stored in the system model. Individual concepts exist for mass budgets, power budgets or the mentioned state machines for example. One of the most substantial aspect of VirSat's customizability is the creation of individual concepts, enabling the user to consider new engineering disciplines in the system model and reuse the concept for subsequent projects if desired. The versatility of the generic elements concepts are built of offers a a high degree of flexibility in invoking new engineering disciplines. The individual concepts or disciplines, enriched with data, then together form an integrated system model, ideally covering all aspects of the system and therefore strongly supporting the idea of MBSE.

Moreover, it is possible to create Java applications in VirSat. With these so called apps the user can access the actual CDM instantiation of each project and furthermore write new information into it. Once created, the apps can be copied to any new project, given that the new project supports all concepts the app accesses. The Java interface also allows to use other functions of Java programming language, e.g. creating and writing files. This can be used to process data and export it to a desired format, as it is done in the course of this work.

The following sections are meant to introduce the reader to the functions offered by VirSat, the structure of the project representation and the process of customization.

Virtual Satellite uses a hierarchical model to represent a project's structure. This manifests in a tree structure of the whole project repository. In the uppermost level of the data structure are by default the role management instance, unit management instance and the repository instance, that contains all available project trees. For storing external documents like specification documents or relevant European Cooperation for Space Standardization (ECSS) standards, every structural element also has a "documents" folder assigned which also exists in the operating system file explorer under the path of the project's repository.

2.3.1 Virtual Satellite Existing Concepts

The fact that VirSat is a tool that relies on customizability does not mean the standard version lacks functionality. It is sufficient to already consider a number of aspects of a system model and, as mentioned, it is continuously expanded. These aspects are represented by some concepts in the standard VirSat version and include product structure, mass budget, power budget, functional electrical architecture, design maturity, requirements, state machines and visualization which are briefly introduced in the following paragraphs.

"Product Structure" is basically the foundation of all concepts. It covers the different product trees that serve as the skeleton for the representation of all aspects of the model. In this concept there are three main trees that can be used, following the maturity of the system composition through the project phases. The first tree to be created is the project tree, in which components to be used are modeled. Afterwards, the configuration tree is created to model the actual configuration of the spacecraft, that is how the actual system implementation shall look like. Finally, the assembly tree represents the configuration tree, enriched with data of the actually assembled components. Thus, the trees of this concept are filled with more and more accurate data during the whole development of a spacecraft. Within the trees, each subsystem has its own branch and each level below represents one more level of detail. The number of levels is not limited [47]. Data can be added for each level from system level to the lowest component level. In general it is favorable to add data in the lowermost possible level to achieve greatest possible detail in the system model. However, not all applications require a high resolution in terms of components, so there are exceptions to this statement.

The ”Mass Budget” and ”Power Budget” concepts enable the user to store component masses and power demands with certain margins and offers pie chart graphics to display the composition of the overall budgets of the system and their margins. Besides giving systems engineers a quick overview over the budgets left for allocation, it also provides transparency to the domain engineers.

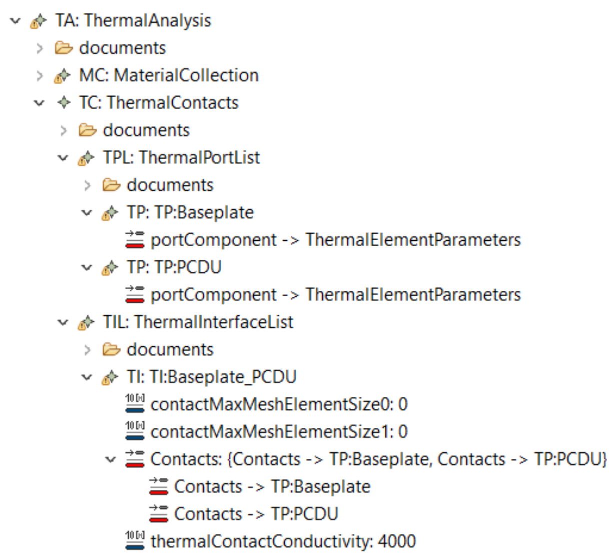
”Design Maturity” adds the possibility to specify the technology readiness level (TRL) of individual component. This way the overall technological maturity of the project can be traced by means of summarizing the individual TRLs.

The ”Requirements” concept adds some complexity as it allows to create certain requirement types with specified attributes. Then, for every desired component a requirement specification is instantiated by the user, that can be filled with actual requirements of the types created before. As of now, this concept is still being developed with the ultimate goal that all requirements and their fulfillment are traced automatically in VirSat whenever possible. Underlining this, as a first step it is currently possible to link requirements to specific properties of components for traceability.

To represent the electrical architecture of the system the ”Functional Electrical Architecture” concept is utilized. It can be used to display the electrical interfaces between components and their characteristics. Analogously to the ”Requirements” concept, one top level element is created, where the types of interfaces are defined. Interface ends of type ”power” or ”data” are added to individual components. These interfaces are then connected with either a power interface or a data interface. Voltage or voltage ranges can be defined in order to specify the interfaces and to track the associated requirements or identify incompatibilities between components. From the defined architecture a diagram showing all interfaces can be created. In contrast it is also possible to create the diagram and let the software translate this to the system model.

Finally, the last concept that is currently usable is the ”State Machines” concept. With this concept it is possible to build state representations with different states or modes of the system. A state machine can be added to each component. To define the possible states of the component, ”states” are added to the state machine. The dependencies between different components and states can be expressed with ”Allows Constraints” and ”Forbids Constraints”. Furthermore transitions between the different states can be modeled in each ”State Machine” instance. As in the concept before, it is possible to generate a state diagram from the defined states and transitions or to translate them from a self created diagram to the system model.

It is visible that all these concepts are strongly linked to the MBSE approach, improving traceability of requirements and compatibility between components in various characteristics of the system. With this being the current state of the engineering data VirSat covers, in the following the ac-



```
TA: ThermalAnalysis
  documents
  MC: MaterialCollection
  TC: ThermalContacts
    documents
    TPL: ThermalPortList
      documents
      TP: TP:Baseplate
        portComponent -> ThermalElementParameters
      TP: TP:PCDU
        portComponent -> ThermalElementParameters
    TIL: ThermalInterfaceList
      documents
      TI: TI:Baseplate_PCDU
        contactMaxMeshElementSize0: 0
        contactMaxMeshElementSize1: 0
        Contacts: {Contacts -> TP:Baseplate, Contacts -> TP:PCDU}
          Contacts -> TP:Baseplate
          Contacts -> TP:PCDU
        thermalContactConductivity: 4000
```

Figure 2.10: Excerpt from a VirSat system model, with the concept described by the code in listing 2.1 is instantiated with the two components "PCDU" and "Baseplate". Note that the "ThermalElementParameters" element, that is referenced in the "ThermalPort" element is in another part of the code and not displayed here. Source: Own representation.

tual structure of the product trees, as well as how the CDM can be complemented by self-created concepts is explained.

2.3.2 Virtual Satellite Concept Development

To add relevant project data that is not covered by existing concepts to the system model, VirSat offers the possibility to create own concepts. Concepts can be seen as specific infrastructure for adding data of a certain discipline to the system model. As mentioned before they exist seperately for different engineering disciplines since most disciplines require to store their own very specific information in the system model. Once created, concepts can be reused for new projects.

To customize the set of concepts, DLR-SC provides an IDE version of VirSat. This version of VirSat enables the user to create individual concepts, using a high level programming language sufficient for creating the required infrastructure to store various types of data. This high level code is then automatically converted to regular Java code in order to enable the use of the concept in the actual VirSat program.

A concept usually consists of multiple elements, where every element is either a category or a structural element instance (SEI). The latter embodies a folder-like object, in which multiple categories (or further SEI) can be instantiated. Categories then embody a collection of properties which host the actual data. An example for a small part of a concept's code is shown in listing 2.1 and its corresponding instantiation in Fig. 2.10.

The *StructuralElement* keyword starts the definition of a new structural element. It is followed by

Listing 2.1: Example for the top level programming language used for creating concepts in the eclipse IDE of VirSats developer version.

```
Concept de.dlr.sc.virsat.model.extension.thermal displayname "thermal" description "Concept for modeling thermal properties"
     version 1.32 beta {

        StructuralElement ThermalAnalysis description "Here all general information for the analysis is stored"{
                Applicable For All;
                Cardinality 1;
        }

        StructuralElement ThermalContacts description "Here all thermal contacts are stored"{
                Applicable For [ThermalAnalysis];
                Cardinality 1;
        }

        StructuralElement ThermalPortList description "Here all thermal ports are stored"{
                Applicable For [ThermalContacts];
                Cardinality 1;
        }

         StructuralElement ThermalInterfaceList description "Here all thermal interfaces between ports are stored"{
                Applicable For [ThermalContacts];
                Cardinality 1;
        }

        Category ThermalPort description "Modeling of thermal connections" {
                Applicable For [ThermalPortList];
                Reference portComponent of Type ThermalElementParameters;
        }

        Category ThermalInterface description "Modeling of thermal interfaces"{
                Applicable For [ThermalInterfaceList];
                Reference Contacts [2] of Type ThermalPort;
                FloatProperty thermalContactConductivity unit "Watt per Meter and Kelvin";
                FloatProperty contactMaxMeshElementSize0 unit "No Unit" default 0;
                FloatProperty contactMaxMeshElementSize1 unit "No Unit" default 0;
        }
```

descriptive properties such as its name and possibly a short version of its name and a description of the element. For each structural element the location where it can be generated in VirSat must be defined. There are three alternatives to achieve this. One is to chose *Applicable For All* which enables the element to be generated directly in the repository, or in every other structural element instance. Secondly, if *Applicable For* is chosen, another structural element name is handed as an argument. The structural element can then only be created inside a structural element of the specified type. Third, *IsRootStructuralElement* can be chosen instead of the *Applicable For* expression. If so, the structural element can only be created in the repository root level. This should not be used extensively, since the root repository can easily become chaotic with increasing amount of root SEIs.

Creating new categories is done with the *Category* keyword. As for the structural element, the headline is completed with the name of the category and optionally a short name and a description. In addition, a category can also inherit from another category, which is also defined in the headline.

The definition of the locations where a category can be instantiated works analogously to the structural elements. Only that categories can only be created within structural elements.

Following on, attributes for the category can be defined from a number of different property types. The properties are used to store and access data bits such as power consumption or mass and can be of string, boolean, float, integer, or list type. The list is used for specifying some predefined values to choose from for this property. A unit can be assigned to all numeric properties, allowing VirSat to convert between different units internally. From the many other attributes that can be defined, the *Reference* attribute is one of the most notable ones. With this, any other structural element, category, or attribute from any other category can be referenced. This enables in VirSat the interconnection between different components for representing interfaces for example. The last attribute to be presented here is the *Resource* attribute. This creates an interface in VirSat that enables to reference or copy an external file from the windows explorer.

2.3.3 Virtual Satellite App Development

Other than the concept development, which is done in the eclipse IDE, apps can be created directly in the VirSat application. They are used to access the system model's data and process it. Thus, as already indicated in the chapters before, apps are of enhanced use since the whole Java Development Kit (JDK) can be invoked for them. In addition, a useful tool for conveniently accessing parts of the system model as a Java bean is provided. This way, one can quickly access certain data of a property, for example a parameter together with its unit can be directly as output using the bean object of it.

Apps are activated and created directly in the ”apps” resource in the project repository. When editing an app, it is opened in a Java editor. The apps are essentially written in Java, such that every app looks like a regular Java program. The system model is by default accessed using the

”ModelAPI” class, which creates an accessible Java object from the VirSat project, that has essentially the same hierarchical structure as the project. The desired properties of the model can then be obtained using the Java specific syntax for navigating through the levels of the model and some functions related to Java beans, allowing to directly access all instantiations of structural elements using their specified name. For example, the *getDeepChildren*() function can be used to collect all structural element instances of a specified type (such as ”ElementConfiguration”, the type of each component's top level element). Thus, no matter how many levels there are in the configuration tree, the Java beans approach allows to do fast system-wide analyses by directly heading for the low level elements, while the default approach navigates from the top level down to each low level element individually. The app development feature as such is strongly related to the MBSE approach, as it allows to import and export data to and from the system model to perform analyses on the system on the one hand, and to import data from documents like requirement tables to the MBSE tool on the other hand.

2.3.4 Additional Virtual Satellite Functions

In the course of its development additional functions were added to VirSat. Two functions, that are used extensively in this thesis, are especially worth mentioning. On the one hand, it is the 3D visualization feature, which consists of a 3*D Viewer*, allowing to visualize a satellite within VirSat, and the ”Visualization” concept, where the data necessary for the 3*D Viewer* to view the components is placed. The concept allows to move components or change their size and form while immediately seeing the changes in the viewer window. This primitive parametric CAD approach allows the users to conveniently create a geometry of the satellite directly in VirSat. It is also possible to import *.stl mesh files as geometry for a component. This however was not entirely functional at the time this paragraph was written, which is why in this thesis the geometry generation directly in VirSat was used. On the other hand, the second important feature is the *Cad Export Wizard*, which wraps the project hierarchy together with the components' geometries in a *.json file that can be read by the Virtual Satellite workbench in FreeCAD.

Chapter 3

Workflow Concept for Thermal Analysis

As stated in chapter 2.1.1 short response times for giving prompt feedback on the impacts of design changes are favorable. This is especially valid for the early design phases, where major changes leading to major impact with the need of quick assessment are more common than in later development phases. Using Virtual Satellite as a tool to track such design changes and store the system model data consequently leads to the demand for a workflow enabling the execution of analyses using system data stored in the Virtual Satellite system model only. As one important aspect of spacecraft design, thermal analysis made available as early as possible in the development would contribute to the efficiency of development. For this, a workflow must be established covering the setup, execution and evaluation of the analysis, being not only of use in the early design phase, but also in the subsequent design phases. This work is dedicated to conceptioning and optimizing such a workflow. The elaborated concept is introduced without much detail in this chapter while the detailed implementation of the workflow is then described extensively in chapter 4. The structure of the workflow can be disassembled into four main parts, that essentially represent the information flow direction, as it is depicted in Fig. 3.1. The first part is where the necessary infrastructure for thermal models is set up in VirSat. Thus, this part is covering the information flow into the workflow or system model, respectively. In the pre-processing part the information flow is from Virtual Satellite to the simulation tool while in the post-processing part it flows the opposite direction eventually feeding the simulation results back to Virtual Satellite.

The chapter starts with a look on existing similar approaches and the specifications for the workflow to be created before the actual workflow concept is presented.

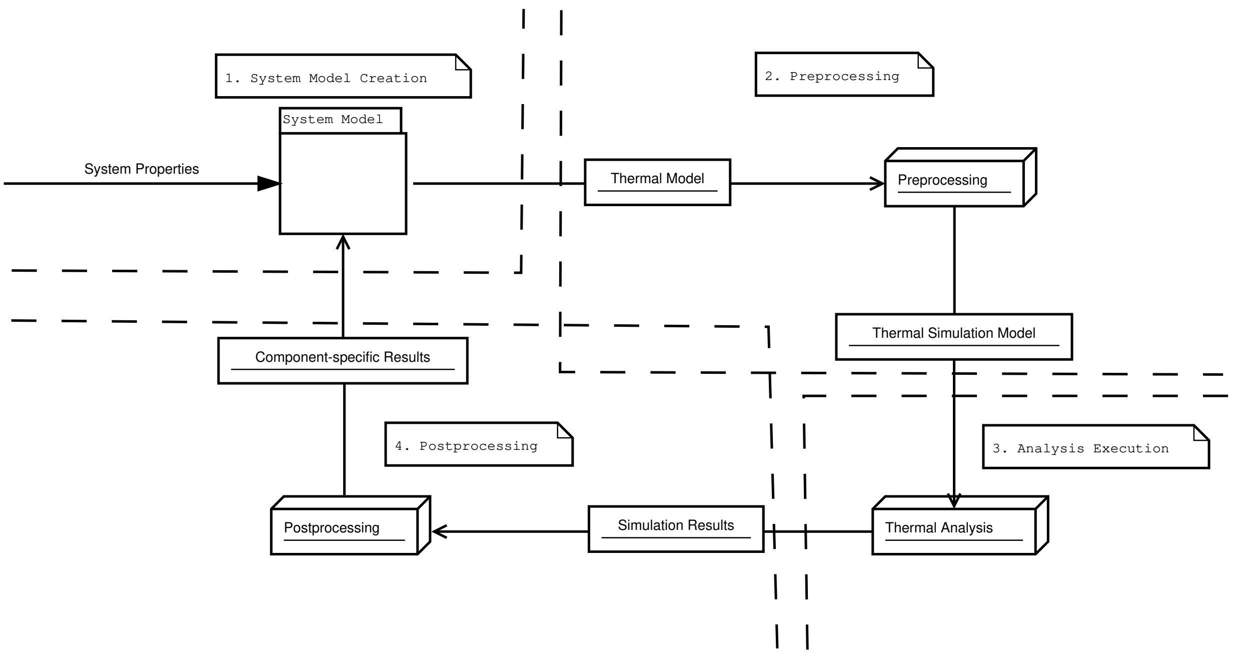


Figure 3.1: Overview of the information flow in the workflow. Source: Own representation.

3.1 Existing Workflows

As the practice of MBSE is still at an early stage of adoption, so far only limited research efforts have been spent on integrating simulation and analysis of physical aspects in MBSE tools. Focus of the resources spent was mostly simulating functional and behavioral system models to verify interfaces and operational modes of a space system [48, 49]. The idea of integrating physical model simulations like mechanical or thermal simulations was also formulated in an MBSE context [13, 50, 51]. However, since functional models have a wider range of application, are by definition well-expressable in system design languages, and are among the very first models built in a project, mostly top level functional and behavioral aspects are respected within the application of MBSE so far. Nevertheless, integrating physical simulations is both necessary and useful. Necessary to fully adopt the MBSE approach on the one hand, useful since it can be a big support for thermal engineers to formulate a proper thermal control concept early on in the development on the other hand. Therefore it is a logical next step in evolving MBSE.

Public research undertakings to integrate thermal simulations into the MBSE approach were executed by Gross et al. as a part of their work developing a unified modelling language (UML) based design language for spacecrafts that covers thermal and mechanical properties among others [52]. Their effort enabled the direct execution of thermal simulation of the modeled spacecraft with the commercial ESATAN software [53]. Some similarities and differences exist between the before mentioned and this work. A major difference is that this work is intended to establish a workflow using the open source software packages VirSat and CalculiX instead of commercial software and that a less wholesome approach is followed, integrating the workflow between two existing tools and only in terms of thermal aspects. The design language used for this will be a slightly customized version of Virtual Satellite's (VirSat) CDM. Still, the general intention of creating physical models using a system modeling language with the least possible effort is one of the main goals of both works.

In the commercial sector, numerous MBSE software packages are available. With focus on requirement management and mentioned top level functional and behavioral aspects, only few of them also include physical analysis capabilities, for example by creating custom flowcharts for an execution sequence of analysis tools like ANSYS [54]. This kind of analysis flowchart design is a similar approach to DLR-Institute of Simulation and Software Technology's (DLR-SC) Remote Component Environment (RCE), where sequences of analyses using different simulation tools can be created and executed [55].

3.2 Workflow Specifications

The workflow to be conceptualized is intended to provide a thermal analysis of a space system, with the least possible effort and as early in the system design as possible. The use of different software packages is required to achieve this; VirSat shall be used as MBSE tool while CalculiX shall be used for the execution of the actual simulation as it was stated in the task description of this thesis. Additional software, such as FreeCAD, may be used as well, if it benefits the quality of the workflow. Factors influencing the quality of the workflow comprise effort of execution, time of execution, proneness to user errors and software prerequisites. When assessing different options, the choice shall be made for the option maximizing the quality of the workflow as it is evaluated by the author.

3.3 Workflow Concept

In this section an overview of the different workflow steps and their function is given. Details on how the steps are executed will not be provided in this section, but in chapter 4. Thus, this section covers the concept of the workflow, while the next chapter is about the actual implementation. With the concept described in the following, the initial workflow concept from Fig. 1.4 can be taken and extended to represent the actual top level concept elaborated in this work in Fig. 3.2.

3.3.1 Virtual Satellite Infrastructure

Keeping in mind that the foundation for MBSE is a consistent system model that includes all relevant data, as a first step it is necessary to create the infrastructure to feed the thermal model data to VirSat. Since so far there is no way of storing thermal properties in the system model other than some mechanical and geometrical properties that are already implemented as part of other engineering disciplines, customizability of VirSat will be used to create the interface needed for

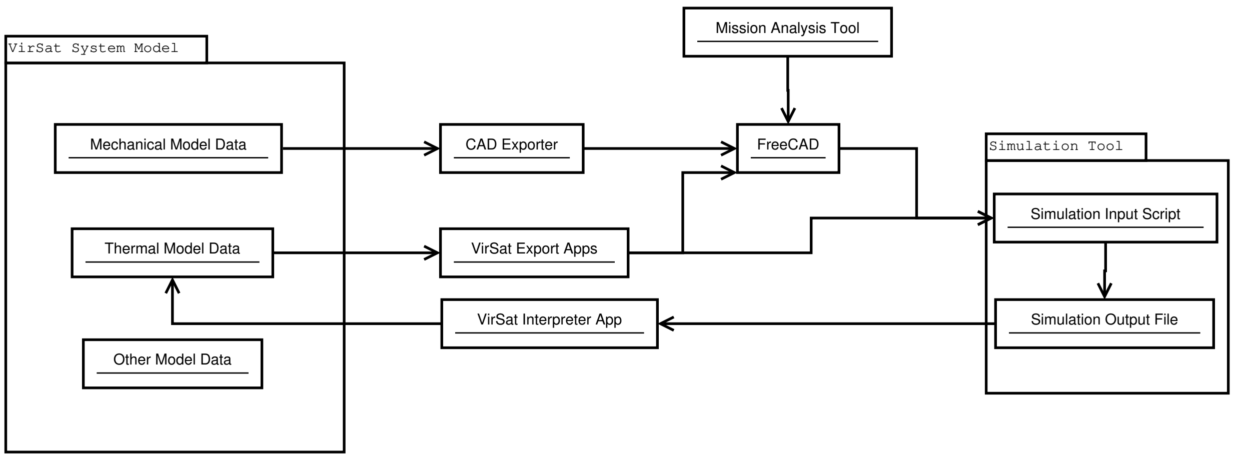


Figure 3.2: Chart of the workflow concept with the tools involved in setting up the simulation and interpret the results. Source: Own representation.

storing all mandatory properties. That means a concept complementing the existing ones with thermal data is created. The different concepts in VirSat are modular, meaning that the amount of dependencies between concepts and especially between specific properties that are likely to be changed in the future shall be minimized. As a result of this even those parameters needed in this work that may already be present in other concepts (or very likely to be included in other concepts in the future) will be part of the new concept, such that the need for other concepts is minimal. When all concepts are sufficiently mature, the double parameters can be reevaluated to be merged into one parameter. Tab. 3.1 lists all basic properties that are used for static and dynamic thermal simulations and thus have to be provided to the simulation tool. The idea is that the thermal concept's structure is divided into two main parts. The "ThermalAnalysis" element, located inside the TCS subsystem instance, hosts all general information about the simulation. The "ThermalData" element in contrast hosts thermal properties of the single components and is therefore located inside their respective components element. The actual implementation of this thermal concept is further described in section 4.2.2.

With the infrastructure for storing the thermal model existing, the model shall then be provided to CalculiX for analysis. However, for executing thermal analyses a number of pre-processing work packages are to be executed first to enable CalculiX to perform a simulation. This includes meshing, assignment of boundary conditions, materials and contacts, and specifying the exact simulation configuration. For the simulation tool an input file must be provided. This file covers all necessary data and is described in detail in chapter 2.2.4.1 and appendix A. The goal of all following pre-processing activities is to achieve a consistent input file with the correct syntax.

3.3.2 Pre-Processing

The above described properties are all stored within the VirSat system model. To access them, the use of VirSat apps is utilized. A number of Java apps is written for exporting the necessary data.

Table 3.1: Thermal data to be provided for a basic spacecraft thermal analysis

Thermal parameter	Description
Absorptivity	Surface-specific efficiency in absorbing radiation
Boundary conditions (heat flow/temperature)	External fixed loads on certain parts of the model
Density	Material-specific/component-specific mass per unit volume
Emissivity	Surface-specific efficiency in emitting radiation
Heat capacity at constant pressure	The amount of heat energy needed for a certain temperature change
Initial temperature	Temperature of the components for $t = 0$
Thermal conductivity	Material-specific ability to transfer heat
Thermal contact surfaces	The surfaces that are involved in each thermal contact
Thermal contact conductivity	Thermal conductivity of face to face contacts between two components
Simulation parameter	**Description**
Time step	For dynamic simulation, the time increment between each discrete step
Total time	For dynamic simulation, the total time determining after how much steps the simulation stops

There is one app for each of the following aspects of the complete model: Material data, thermal contact data, radiative property data, volume heat fluxes, face heat flux among temperature boundary conditions, and finally general simulation data. The Java interface allows to access and directly write the desired properties to text files. Since the input script is a text file, it is favorable directly generating usable input syntax with the apps whenever possible. However, all model properties must be assigned to the right subsets of the discretized version of the model, the mesh, which is not yet known. As a consequence, additional pre-processing steps for correctly assigning the properties to the meshed model are necessary, once the mesh is generated.

Regarding the geometry of the model, the dependence on another concept is inevitable. The ”Visualization” concept offers the infrastructure for creating or importing geometries in VirSat, which is mandatory for simulation setup. The MBSE software also offers a geometry export function, which is complemented by the Virtual Satellite toolbox in FreeCAD, allowing to export the geometric model from VirSat and display it in FreeCAD. It was originally created for changing the model's geometric features in a CAD program and afterwards reexporting the changed geometry from the CAD program back to VirSat [56]. With FreeCAD also offering a workbench with functionality for FEA and a Python console, it would be an option for the subsequent pre-processing steps. A second option besides FreeCAD is Gmsh. Gmsh is a meshing tool that also provides CAD functions besides also having scripting capabilities. Gmsh is also one of the two tools utilized by FreeCAD for meshing geometries, the other being Netscape. Nevertheless, due to the fact that Gmsh is not able to read the exported *.json geometry file, FreeCAD must be employed at this point in any case. In addition, the integrated Python console, which offers tools to perform tasks on the geometry, and the integrated mesh tools offer some of the functions necessary for further pre-processing. Thus, it is a very versatile software for pre-processing the thermal analysis model. Consequently, although the documentation of its FEA workbench is not yet complete, FreeCAD was chosen for conducting the subsequent pre-processing steps.

3.3.2.1 FreeCAD Pre-processing

With the pre-processing tool being selected the part of the workflow using FreeCAD is to be eased as far as possible. Inevitably, this means that a script for the Python console is written, performing as many actions as possible automatically. This reduces the time of execution of the workflow on the one hand, and the risk of errors produced by the user on the other hand. Thereby increasing the overall quality of the workflow in two ways.

The list of tasks to be executed before the model is completely processed for the simulation is still quite long, including

1. preparing geometric model for assignment of face-specific properties in VirSat
2. mesh generation
3. mesh group generation for assignment of properties

4. assignment of thermal load boundary conditions and radiative properties (that are not assigned with a VirSat app
5. mesh refinement on contact areas
6. processing and assignnment of external input (e.g. Sun vectors to resulting radiation intensity).

Keeping in mind that the number of different software packages used should be as low as possible, the ideal case is to complete all these tasks using FreeCAD only.

With the geometric model imported in FreeCAD, it is possible to directly create a finite element mesh exclusively using console commands. This enables the use of scripting for fast mesh generation. With the function to access also parts of the mesh with the Python console, the relevant nodes and elements that the thermal properties address can be isolated and added to groups, to which then the properties are assigned. Ensuring the naming of the groups follows a convention, the properties can be assigned by a VirSat app to group names which not yet exist, but are certain to be created by FreeCAD later on. This means the actual assignment of most properties is done directly with the Java apps, while the related groups are then defined using FreeCAD (see Fig. 3.3). The pre-processing using the FreeCAD script, and all functions related to it are described in more detail in section 4.3.

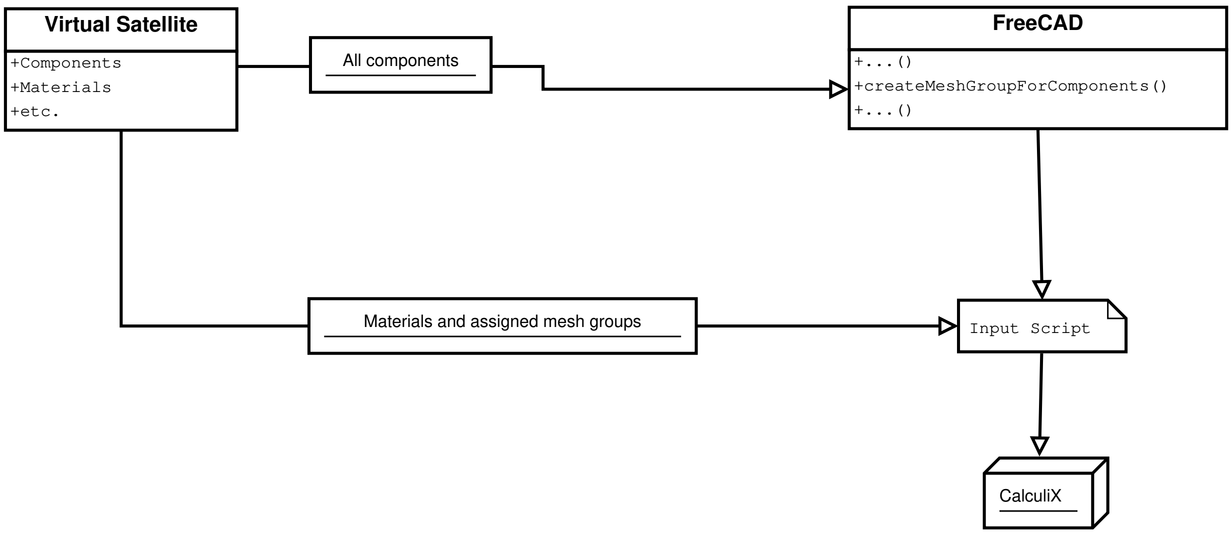


Figure 3.3: Separation of the actual mesh group generation and the assignment of the materials to the mesh groups. Materials are assigned in the input script to mesh groups that are generated in a subsequent step. Source: Own representation.

3.3.3 Inclusion of External Factors

Since in space the thermal behavior of a spacecraft is strongly influenced by the thermal environment in its orbit, it is beneficial for the validity of a thermal analysis to take external factors into consideration. As depicted in Fig. 2.3 these are mainly solar radiation, Earth reflected solar radiation and Earth infrared radiation. To account for solar radiation as the first and most significant contribution to the thermal load on the spacecraft model, it is necessary to obtain the vector of the Sun with respect to the spacecraft and, if a dynamic simulation is executed, the course of the vector over time. To determine this, the use of external tools is inevitable, since the determination of orbit position and attitude would exceed the scope of this work. Thus, the mission analysis tool Systems Tool Kit (STK) is used to obtain the Sun vector, the Sun intensity regarding the obstruction of the Sun by Earth, the Earth vector and the angle between spacecraft, Earth surface and Sun for the specified time steps.

The obtained orbital data is then automatically read and interpreted in the FreeCAD script to combine the loads for each face and assign them. This is described in section 4.3.8 Generally, it does not matter which tool is used for generating the files, as long as the data is provided as a comma-separated values (CSV) file with the according structure presented in appendix B.6.

3.3.4 Post-Processing

After completing the input script, the simulation can be executed. CalculiX offers a post-processor that can be used to view the results of the simulation. Still, the idea of MBSE is to obtain the simulation results in the MBSE environment. To achieve this, the output file of CalculiX must be interpreted such that the desired values can be added to the system model. Interpreting is done with another Java app that reads the output file and finds the right locations in the file for obtaining the correct values. Primarily this concerns maximum and minimum temperature in the simulation for every component. To enable requirement traceability, the mentioned maximum and minimum temperature are added to a result subset of the system model, where the corresponding requirements can be linked to. By comparing the required temperature with the obtained one, fulfillment of the requirement can be evaluated. In addition, for respecting requirements regarding the temporal change of temperature, Virtual Satellite's (VirSat) default equation feature can be used together with the defined timestep duration values to calculate the temperature change for each timestep.

Chapter 4

Implementation of a Workflow for Thermal Analysis

The before stated concept for a workflow to conduct thermal analyses, using VirSat as starting and ending point, provides a top level overview of the data that is transferred and where it is forwarded from/to. However, to be able to execute this workflow, a far more detailed view on the actual actions is necessary. Aspects that seem to be trivial from a top level view can quickly become extensively time-consuming when looking into the very detail. After all, this work is not only about stating the workflow itself, but to optimize it and enable its use as well.

The quality of a workflow is greatly influenced by the ease of execution and the effort for execution. Or generally spoken, its susceptibility to errors and its effort of execution. As humans are prone to errors and need longer for certain, especially repetitive tasks, a focus of this work is not only to execute the workflow, but also to automate it whenever possible.

This chapter is intended to present the workflow and a detailed description of the processes and automated functions utilized to achieve the overall goal of the simulation round trip. It does so by first presenting how the actual structure of the input files in this work looks like. Large parts of this chapter and of the workflow in general are then dedicated to achieve a representation of the thermal model in this input format. Finally, the chapter is completed by describing how the output of CalculiX is returned to the system model in VirSat.

4.1 CalculiX

In this workflow, CalculiX has an important role. However, since the only task is to execute the analysis, there is not much to tell about it in this chapter. The general input file structure is described in section 2.2.4.1 and A, but there is an addition to this structure. This addition is strongly linked to the implementation of the workflow. Thus, it is discussed here.

4.1.1 Input File Separation

The composition of the input file is discussed in section 2.2.4.1 and appendix A. However, an important addition is that there is a keyword that allows to split the input file into an essentially arbitrary number of input files. This is done with the *Include* keyword, which invokes another file in the same folder with a given name.

Without this function, the input file would become extensively long and thus, very hard to oversee. Furthermore, it would be necessary for every single application that adds or changes something in the input file to open it, find the right location inside the file and write the information correctly. Keeping in mind, that there are several thousand of nodes and elements at least, it is not optimal if every application is required to iterate through the whole file. Moreover, this is anticipated to be quite prone to errors. For example it might be possible that the application confuses the appropriate locations to write to and, by crashing the syntax, essentially rendering the file unusable. Furthermore, it is hard to find the location where the erroneous text was inserted as the file is extensively long. On the contrary side, depending on the number of components, the use of this function can lead to quite a large number of separate files,that are all necessary for the simulation.

Still, the modular multi-file approach was chosen as it offers less complexity when it comes to generation and more variability when it comes to manual adjustments like removing single components from the input deck by hand. Moreover, with this approach the actual order in which the single files are generated does not matter, which also decreases the risk of errors. Finally, it is faster to track errors in the input deck with this approach.

Listing 4.1 shows the main input file that invokes all individual component- and property-specific input files that exist for the thermal model. In the following, the whole process of obtaining these individual input files is described, which is the ultimate goal of all pre-processing activities that follow. For this the first aspect is to cover all activities in VirSat, as this is the starting point where the system model that is to be transformed into a thermal model for analysis is instantiated in.

4.2 Virtual Satellite

Since Virtual Satellite is the source of all data used for setting up the simulation it is the first link in the chain of elements of the workflow. Still, data source does not sufficiently sum up the role of Virtual Satellite in the simulation setup flow. In fact the tools available to customize Virtual Satellite are used extensively. Not only to tailor Virtual Satellite according to individual needs of the thermal system but also to supply some of the data in that way, that it can be directly read by the simulation program. To achieve the latter the Virtual Satellite Apps feature, as it is described in chapter 2.3.3, is utilized. Offering essentially all Java functions, it is a versatile tool that can be used for a wide range of tasks. However, the first step in this section is to feed system data to Virtual Satellite such that the Java apps can access it.

Listing 4.1: Example of a main input file invoking a number of other input files.

```
*INCLUDE, INPUT=Baseplate_2c06530c_4ee1_4388_bff8_2e55afaae114.inp
*INCLUDE, INPUT=PCDU_e7ca332f_4740_4170_b9ce_1029ee436162.inp

*PHYSICAL CONSTANTS, ABSOLUTE ZERO = 0, STEFAN BOLTZMANN = 5.669E-8

*INITIAL CONDITIONS,TYPE=TEMPERATURE
Nall,293

*INCLUDE, INPUT=Materials.inp

*INCLUDE, INPUT=add_contact.inp
*INCLUDE, INPUT=contact_surfaces.inp

*STEP

*HEAT TRANSFER, STEADY STATE

*DFLUX
*INCLUDE, INPUT=Baseplate_2c06530c_4ee1_4388_bff8_2e55afaae114.bfl
*INCLUDE, INPUT=Baseplate_2c06530c_4ee1_4388_bff8_2e55afaae114.hf
*INCLUDE, INPUT=Baseplate_2c06530c_4ee1_4388_bff8_2e55afaae114.bc


*DFLUX
*INCLUDE, INPUT=PCDU_e7ca332f_4740_4170_b9ce_1029ee436162.bfl
*INCLUDE, INPUT=PCDU_e7ca332f_4740_4170_b9ce_1029ee436162.hf
*INCLUDE, INPUT=PCDU_e7ca332f_4740_4170_b9ce_1029ee436162.bc


*RADIATE
*INCLUDE, INPUT=Baseplate_2c06530c_4ee1_4388_bff8_2e55afaae114.rad
*INCLUDE, INPUT=PCDU_e7ca332f_4740_4170_b9ce_1029ee436162.rad

*NODE FILE
NT
*EL File
HFL
*END STEP
```

4.2.1 Storing Model Data

As stated in Tab. 3.1 a thermal analysis needs various bits of information as input for being able to deliver a valid result. They can be divided into four main types: Geometrical data, material data, boundary conditions and configuration data.

Geometrical data describes the dimensions and shape of the involved bodies. Material parameters describe the relevant physical behavior of the materials for a thermal analysis. Boundary conditions are the temperature, heat flow, radiation and thermal contact data used for the simulation. Configuration data mostly includes general, non model-specific information necessary to set up the simulation. Except for geometrical data, all of this information is quite sophisticated and therefore mostly not yet considered in the VirSat conceptual data model (CDM). To store this data a new concept for Virtual Satellite, called ”thermal concept” hereinafter, was created in the scope of this work. This allows to store necessary information bits in the CDM.

The concept was created using the Virtual Satellite Developer IDE Release 4.12.0. The integrated development environment offers the possibility to create a concept on a rather high level programming language with basic functionality allowing to create infrastructure to store data. The code is then processed and translated to Java code to be integrated into the regular VirSat source code. Concepts consist of two types of elements that can be put into relation with each other, the first one being a ”structural element”. Structural elements can be considered as containers that host either so called categories or additional structural elements. On the one hand, structural elements can be located in the root tree where they represent some general top level information that is not linked to the different product structure trees. On the other hand, they can also be located in every lower instance, if they are intended to be linked to a particular component or subsystem for example. Categories in contrast are always located inside structural elements. The very structural element where a certain category is supposed to be located can be, but does not have to, precisely specified by its name. If not specified, the category can be instantiated within all structural elements. The categories again host alphanumeric properties which are used to store specific data bits. An excerpt of a VirSat project with structural elements, categories and properties marked is provided in Fig. 4.1.

Another feature of categories are references. They enable the user to reference objects of another category or even entire categories or structural elements inside a category. This is especially useful if certain connections between elements need to be expressed in the data model, for example to model interfaces. The single elements and the structure of the thermal concept are addressed in detail below.

4.2.1.1 ImportMaterials

For the sake of easing the modeling of in new projects, an app was created that reads a CSV file containing material data. This material data is then placed inside the ”MaterialCollection”, that is further described in section 4.2.2.6. This way, a project-overarching material library can be cre-

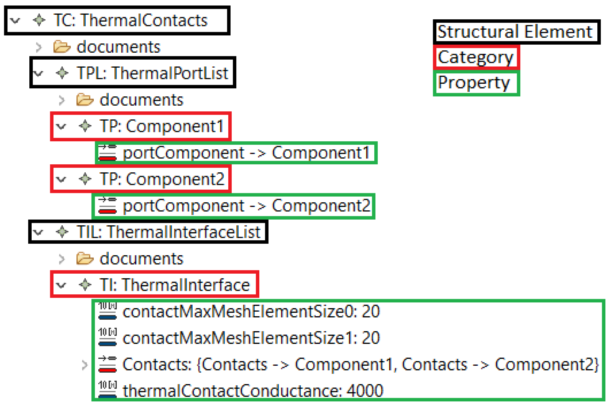
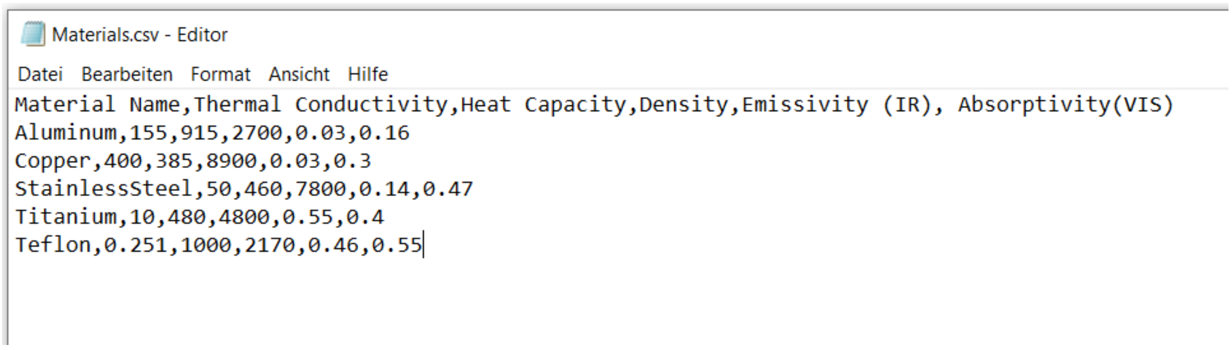

TC: ThermalContacts
documents
TPL: ThermalPortList
documents
TP: Component1
portComponent -> Component1
TP: Component2
portComponent -> Component2
TIL: ThermalInterfaceList
documents
TI: ThermalInterface
contactMaxMeshElementSize0: 20
contactMaxMeshElementSize1: 20
Contacts: {Contacts -> Component1, Contacts -> Component2}
thermalContactConductance: 4000
Structural Element
Category
Property

Figure 4.1: Excerpt from a system model. Structural elements are marked with a black box, categories with a red box and properties with a green box. Source: Own representation.


Materials.csv - Editor

Datei Bearbeiten Format Ansicht Hilfe

```
Material Name,Thermal Conductivity,Heat Capacity,Density,Emissivity (IR), Absorptivity(VIS)
Aluminum,155,915,2700,0.03,0.16
Copper,400,385,8900,0.03,0.3
StainlessSteel,50,460,7800,0.14,0.47
Titanium,10,480,4800,0.55,0.4
Teflon,0.251,1000,2170,0.46,0.55
```

Figure 4.2: Example file for storing material data for import in VirSat. Source: Own representation.

ated and maintained. For each material in the file, the according name, thermal conductivity in $\mathrm{W\,m^{-1}\,K}$, specific heat capacity in $\mathrm{J\,kg^{-1}\,K}$, density in $\mathrm{kg\,m^{-3}}$, visible infrared emissivity, as well as visible radiation absorptivity must be specified. An example for such a file is depicted in Fig. 4.2. To avoid misinterpretations and other issues potentially occurring later in VirSat the material names shall not have blankspaces.

4.2.2 Thermal Concept Structure

In the following, the created thermal concept is presented and its structural composition is explained in detail. The concept consists essentially of two main parts, the structural elements "ThermalAnalysis" and "ThermalData". "ThermalAnalysis" is a structural element to contain information regarding the thermal simulation as such which hosts the structural elements "BoundaryConditions", "MaterialCollection", "MeshSizes", "ThermalAnalysisResults", and "ThermalContacts" as well as the category "AnalysisType". Each of these structural elements again hosts structural elements or categories. The "ThermalAnalysis" structural element is meant to be located inside the thermal control subsystem structural element of the configuration tree, however it can actually be created in every single structural element due to the fact that it is not possible to specify a spe-

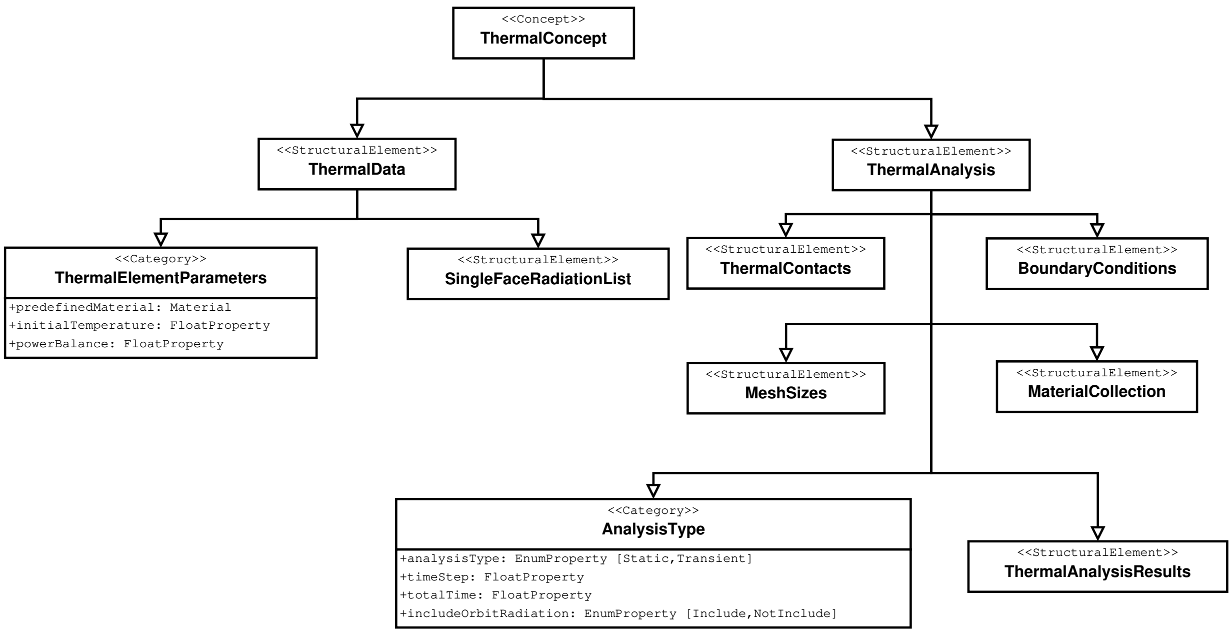


Figure 4.3: Diagram showing the structure of the thermal concept's first two levels. Source: Own representation.

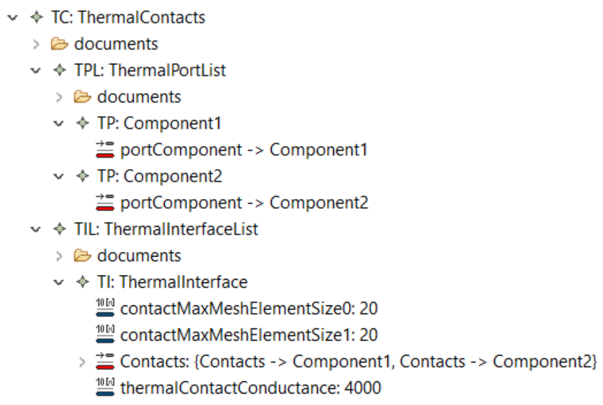
cific type or name of structural elements where it can be created. Although other locations would most likely not cause any problems, it is highly recommended to the user to follow this topology. The above named elements host mostly general thermal model data, that is linked more to the thermal model than to the individual components themselves.

In addition to the "ThermalAnalysis" element, the concept consists of the "ThermalData" element, which is dedicated to component specific data. This structural element is intended for the lowest possible level components of the configuration tree since it hosts data that increases the detail and quality of the simulation, the lower the level of the described component is.

The "ThermalData" element hosts the "ThermalElementParameter" category, as well as the structural element "SingleFaceRadiationList", which are presented in detail among the other elements in this section. For a lucid overview of the concept's structure, Fig. 4.3 may be considered.

4.2.2.1 "ThermalContacts"

An important aspect in modeling the thermal behavior of a system, especially a space system which is not influenced by convection, is considering the contacts between the single components. To account for this, the "ThermalContacts" structural element was created. It serves as a container for the further structural elements "ThermalPortList" and "ThermalInterfaceList". The "ThermalPortList" is intended to host a port for every component in the model. For every port in the list the according component must be referenced in its properties in order to be defined properly. Fig. 4.4 provides a view of the corresponding part of the system model and the relevant properties that can be implemented with this part of the thermal concept.



```
TC: ThermalContacts
  documents
  TPL: ThermalPortList
    documents
    TP: Component1
      portComponent -> Component1
    TP: Component2
      portComponent -> Component2
  TIL: ThermalInterfaceList
    documents
    TI: ThermalInterface
      contactMaxMeshElementSize0: 20
      contactMaxMeshElementSize1: 20
      Contacts: {Contacts -> Component1, Contacts -> Component2}
      thermalContactConductance: 4000
```

Figure 4.4: View on the implementation of the part of the system model where the thermal contacts are defined. Source: Own representation.

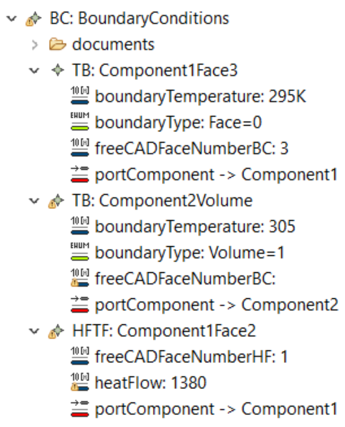
The "ThermalInterfaceList" lists all existing contacts between components. For this, the properties of every interface in the list comprise the reference to two thermal ports as well as a value for the thermal contact conductivity characterizing the thermal interface. It is important to point out that the first component named will be treated as the master component. In fact this only has an influence if there is a one-dimensional contact somewhere in the model, that shall be treated as a two-dimensional contact. If this applies, the components must be slightly shifted into each other, such that they are slightly overlapping, depending on how much actual contact area is desired. The master contact component will then cut the overlapping volume out of the slave contact component in order to obtain a consistent thermal model. For regular area contacts that do cover an area greater than zero, master or slave contacts have no influence.

In addition, the user is enabled to set mesh sizes for each of the two involved bodies. If it is set to zero, it has no influence. Any value above zero will set the local mesh element size at the contact area to the specified value as maximum. Thus, for critical contacts the mesh can easily be locally refined with this option. The first element size with the 0 identifier relates to the master component, while 1 relates to the slave component.

4.2.2.2 "BoundaryConditions"

Boundary conditions are very important for thermal analyses and represent external loads or fixed values. Not all boundary conditions for thermal analyses are covered in this element, only the ones that are not covered in other categories and that are applicable for a space system at all. Convection boundary conditions are neglected in this work, as mentioned in section 2.2. Volumetric heat flow, meaning the thermal power generated by the component itself, and radiation are considered as an internal component parameter and are therefore contained in the "ThermalElementParameters" in section 4.2.2.5. Therefore, only face and component temperatures as well as face heat flow conditions are considered in this section.

To model the boundaries, the structural element "BoundaryConditions" is added to the "Thermal-



```
BC: BoundaryConditions
  documents
  TB: Component1Face3
    boundaryTemperature: 295K
    boundaryType: Face=0
    freeCADFaceNumberBC: 3
    portComponent -> Component1
  TB: Component2Volume
    boundaryTemperature: 305
    boundaryType: Volume=1
    freeCADFaceNumberBC:
    portComponent -> Component2
  HFTF: Component1Face2
    freeCADFaceNumberHF: 1
    heatFlow: 1380
    portComponent -> Component1
```

Figure 4.5: View on an implementation of the part of the system model where the boundary conditions are defined. Source: Own representation.

Analysis" structural element. Within "BoundaryConditions" there are two categories that can be created. On the one hand, for every temperature boundary condition a "TemperatureBoundary" category is created. Inside this category it must be specified if the condition is a face or volume condition. In addition to this, the actual temperature, a reference to the corresponding component, and, if applicable, the relevant face number in the pre-processed FreeCAD model is stored. On the other hand, for every face heat flow boundary a "HeatFlowToFace" category is created where the heat flow per unit area, a reference to the component and the number of the face in the pre-processed FreeCAD model is supplied.

A view on an implementation of this boundary condition element with one boundary condition of each type as an example is provided in Fig. 4.5.

4.2.2.3 "AnalysisType"

There are two main types of thermal simulations considered in this work; steady state and transient. Steady-state thermal simulations regard a non dynamic state, meaning an equilibrium, where time has no influence on the temperature. This is often used to determine the hot and cold case temperatures of the spacecraft [17]. Transient thermal analyses on the other side display the course of the temperature over time. The results are most important for very precise instruments or experiments, as the rate of temperature change can largely influence the component, for example in the atomic clock mentioned in section 2.2. Moreover, dynamic simulations can be considered if the most probable prevalent temperatures are to be determined. For this kind of simulation a timestep and a total duration need to be defined. To cover these aspects the "AnalysisType" category is introduced. It is placed inside the "ThermalAnalysis" structural element due to its nature as general simulation setup category. Inside the category the properties "analysisType", "timestep" and "maxTime" were created to store the mentioned aspects. The two last mentioned are only necessary when conducting a transient analysis is desired. Otherwise they can be left empty.

4.2.2.4 ”MeshSizes”

In numerical analysis, mesh size always plays an important role for the quality of a result. To account for this, the ”MeshSizes” structural element is used. Here, individual mesh sizes for each component can be specified. This is done by adding ”ComponentMeshSize” categories for each component to the structural element and assign a mesh size to the ”maximumCharacteristicMeshLength” parameter. In addition, the desired component is linked to the mesh size by choosing its ”ThermalElementParameter” instance.

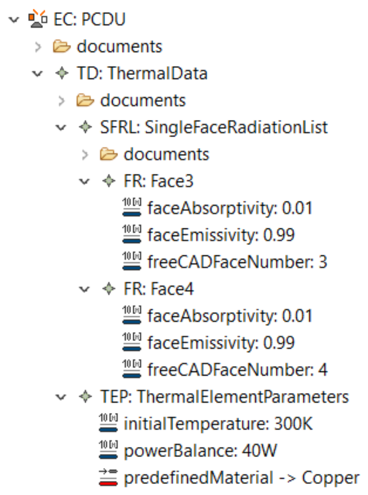
4.2.2.5 ”ThermalElementParameters” and ”SingleFaceRadiationList”

The most important data when setting up a thermal simulation are material parameters, and, especially when conducting a simulation considering radiation as main heat transfer mechanism, the optical surface parameters. To store the parameters a category called ”ThermalElementParameters” is created within the general component-specific structural element ”ThermalData”. The category hosts a property for each parameter needed. Originally, it was intended to be applicable for element configurations only, but for the sake of modularity and independence of other concepts, it is applicable for all structural elements. This decision was also made considering an advice from the developers that there might be future changes in the naming of certain structural elements. Thus, for robustness to potential changes of the structural elements in future versions, it would be safer to use the applicable for all option. Consequently, the thermal engineer has to make sure that the category is not assigned to instances that are not physical parts and have no thermal parameters.

The properties implemented are mostly of numerical nature with thermal conductivity, heat capacity and density being material related quantities. Additionally, the surface properties absorption coefficient and emission coefficient are part of the category.

Although they are technically boundary conditions, the radiation properties are located inside the material specification, as they are material-specific boundary conditions. Two aspects regarding these surface parameters shall be covered in more detail, as they display some important contexts with CalculiX.

On the one hand this concerns the surface itself - not all components have a homogeneous surface throughout the whole body. Indeed, in many cases different surface characteristics are used to influence the thermal behavior of a component or a system of components, as the use of surface mirrors, black paint, or similar indicates [35]. Covering this aspect is important for the usability and validity of this workflow. However, since in VirSat the lowest geometrical representation in VirSat is the complete component shape and not single surfaces, another functionality was added to the thermal concept. To assign radiation parameters to individual surfaces, it is possible to add a ”SingleFaceRadiationList” object to the thermal data instance in which single face instances can be defined. For each single face object the emissivity, absorptivity, and the face number of the pre-processed FreeCAD geometrical model must be provided. Each surface of a component that is



```
EC: PCDU
  documents
  TD: ThermalData
    documents
    SFRL: SingleFaceRadiationList
      documents
      FR: Face3
        faceAbsorptivity: 0.01
        faceEmissivity: 0.99
        freeCADFaceNumber: 3
      FR: Face4
        faceAbsorptivity: 0.01
        faceEmissivity: 0.99
        freeCADFaceNumber: 4
    TEP: ThermalElementParameters
      initialTemperature: 300K
      powerBalance: 40W
      predefinedMaterial -> Copper
```

Figure 4.6: View on an implementation of the part of the system model where the component thermal parameters are defined. Source: Own representation.

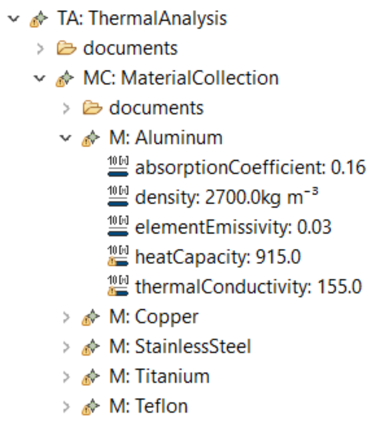
not defined individually with a face emissivity object automatically gets the overall emissivity and absorptivity defined in the thermal element parameters of the whole content.

On the other hand, CalculiX only uses the emissivity to perform its simulation. According to Kirchhoff's law, the absorptivity is equal to the difference between one and the emissivity value [24]. Nevertheless in the analysis of space systems this is not necessarily true, because often the emissivity is considered in the infrared wavelength range, which is the range in which components usually emit most of their radiation according to their operating temperature [17]. Absorptivity is given as the absorptivity in the visible light wavelength, which is the wavelength where the Sun emits most radiation and which mainly influences the spacecraft [17]. The latter is basically ignored in CalculiX since there is only one parameter that can be handed over. The only reasonable one for this intention is the infrared parameter, as CalculiX is supposed to calculate the thermal radiation exchange between the components.

Still, the Sun's radiation cannot be ignored as it greatly influences the thermal behavior of a system as well. Now, to actually account for it, the solar constant is basically used as a distributed flux on the faces and the absorptivity provided in VirSat is used as a factor to multiply by the solar constant for each face. This topic is covered in section 4.3.8.

In addition to the material and surface parameters, there are element quantities that are related to the component as such. These are the initial temperature and the power balance of the component. The set of data is completed by the only string property, the material name of the component. This is necessary since CalculiX assigns a name to each material which must not be empty.

Note that in the current version, some thermal element parameters are outsourced to the "Material" category. Instead of always specifying all those parameters, the user chooses the according material from the "MaterialCollection" specified in section 4.2.2.6. The material then represents these parameters. An example how the "ThermalData" instance for one component could look



TA: ThermalAnalysis
documents
MC: MaterialCollection
documents
M: Aluminum
absorptionCoefficient: 0.16
density: 2700.0kg m^{-3}
elementEmissivity: 0.03
heatCapacity: 915.0
thermalConductivity: 155.0
M: Copper
M: StainlessSteel
M: Titanium
M: Teflon

Figure 4.7: View on the implementation of the material collection in a system model. Source: Own representation.

like with this applied, is provided in Fig. 4.6.

4.2.2.6 ”MaterialCollection” and Import of Materials

The ”MaterialCollection” was established at a later phase of this work. Its intent is to ease the creation of new components in a project. The way it works is that there is one dedicated element for storing all materials with their specific parameters. When creating a new component one only has to choose the material with a reference instead of typing in the parameters. In addition, an app is provided in section 4.2.1.1 for importing materials from a CSV file, such that material libraries can be used over the course of multiple projects while being maintained and consistently extended.

For this, single ”Material” categories can be created inside the ”MaterialCollection” structural element, which is located in the general ”ThermalAnalysis” element. Inside the ”Material” category, the thermal conductivity, absorption coefficient (visible), emission coefficient (infrared), heat capacity, and density parameters are specified for each material as depicted in Fig. 4.7. This way, each material can be stored easily, while the radiation parameters can still be adjusted for single faces, independent of the material, if desired. The inclusion of the general radiation parameters in the material parameters indicates that the surface of the material is included, meaning that the material name might contain an indication of the surface quality (polished, raw, etc.), if it is of importance.

4.2.2.7 ”ThermalAnalysisResults”

The ”ThermalAnalysisResults” structural element is instantiated inside the ”ThermalAnalysis” element and hosts all analysis results that are imported in the course of the simulation results interpreter app described in section 4.4. In the ”ThermalAnalysisResults” one ”AnalysisResult” structural element is created each time the *ReadSimulationOutput* app is executed. Inside the ”AnalysisResult” element there is one ”ComponentResult” category for each component in the thermal model. The maximum and minimum temperature values are stored as properties in these categories. In addition, there are properties to specify the number of the timestep where the according values are reached and property to reference a ”ThermalElementParameter” element of a component. These two properties are not yet used by the app but could be added in the future.

4.2.3 Accessing and Exporting Model Data

4.2.3.1 Accessing Model Data

Having established a new infrastructure to store all material, boundary and simulation data, the next step towards a thermal simulation is exporting that data from the storage location, in this case Virtual Satellite, to the correct format and location it needs to be to set up the simulation. As mentioned before geometric data storage and export is already implemented in Virtual Satellite. Consequently, the geometry is exported using the dedicated VirSat *Cad Export Wizard*, meaning no application has to be developed for it. Thus, geometric data is widely ignored in this subsection. For all other data types the means to export them need to be created in the course of this work. This is done using the Java programming environment and an application programming interface (API) provided by the App function of Virtual Satellite. A glimpse on the data storage structure is already provided in sections 2.3.3 and 4.2.1. However, it follows the same hierarchical structure as the elements in the regular project view of VirSats GUI. As an example, to access the thermal conductivity of a component one needs to navigate through the system hierarchy starting with the root structural elements. First, the configuration tree must be selected, to access what is inside. To select the right structural elements, the *getRootSeis*() command is utilized with the configuration tree class as an argument. This way a list with all configuration trees is obtained from which the desired one is to be used. Now, to get to the component level the subsystem containing the component is selected with a *get*() command. With the *getChildren*() command and the element configuration class as argument all components within the subsystem are listed. To get to the thermal element parameters the specific component is chosen and with the *getFirst*() command and the thermal element parameters class as argument one has arrived at the level containing the actual parameters. To get the desired parameter, in this case the thermal conductivity, commands are supplied for every property within the thermal element parameters class. The thermal conductivity can now be obtained with *getThermalConductivity*(). An example for this is shown in listing 4.2

This is a precise but somewhat exhaustive way of obtaining specific property values. Another possibility is to make use of the *getDeepChildren*() command. With this it is possible to obtain all child elements (only categories) and further child elements of a certain structural element type including all structural elements that have the desired data. From that set the right subset can be obtained by iterating through it with a certain conditional statement to check if each element belongs to the desired subset.

Listing 4.2: Example app for obtaining the thermal conductivity from Virtual Satellite.

```
public class AppExample1 {
        public static void main(String[] args) {
                ModelAPI modelAPI = new ModelAPI();
// get all configuration trees
                List<ConfigurationTree> cts = modelAPI.getRootSeis(ConfigurationTree.class);
//select first configuration tree
                ConfigurationTree ct = cts.get(0);

//select all subsystems (within the configuration tree
                List <ElementConfiguration> ecs = ct.getChildren(ElementConfiguration.class);
//select second element configuration (subsystem)
                ElementConfiguration ec = ecs.get(1);
//select all components within subsystem
                List <ElementConfiguration> elcs = ec.getChildren(ElementConfiguration.class);
                ElementConfiguration elc = elcs.get(0);
//select all thermal data (technically there shall be only one)
      //select first component
                List<ThermalData> tds = elc.getChildren(ThermalData.class);
//select the first (and only) thermal data element
                ThermalData td = tds.get(0);
                             //select first thermal data
//obtain the referenced material in this thermal element parameters instance
                Material mat = td.getFirst(ThermalElementParameters.class).getPredefinedMaterial();
//obtain thermal conductivity from this material and convert the value to base units
                double thermalConductivity = mat.getMthermalConductivityBean().getValueToBaseUnit();

                System.out.println(thermalConductivity);
        }
```

4.2.3.2 Exporting Model Data

After accessing the data model as it is described in section 4.2.3.1, the next step is to export it to a format such that CalculiX interprets it correctly. As this subsection largely focuses on how to arrange all data to match the input format of CalculiX, the reader might consider reading section 2.2.4.1 and appendix A first, to have in mind how such an input script looks like. To avoid errors and ensure valid simulation results, the exported data must match the CalculiX input structure syntactically and correctly represent the thermal model. As described in chapter 4.1.1 the actual sequence in which the files are generated does not make a difference in the selected approach. This allows to create a dedicated app for the export of every type of information, enabling the possibility to quickly change various values in VirSat without having to export the whole set of data which might also contain changes which are not yet desired to be fed to CalculiX. The following subsections briefly address the actual implementation and the logic of the single export apps. Details about the actual structure of the input files created can be looked up in sections 2.2.4.1 and 4.1.1. Generally, the data is either exported directly to the CalculiX input format, or to an intermediate format for the FreeCAD script to open and process it further, as it is depicted in Fig. 4.8. Ultimately, the goal is to provide the complete simulation model in the input format to execute an analysis with it. An overview of the aspects of the thermal model to be exported from VirSat, the app that exports it and the format they are after the corresponding app was executed is provided in Fig. 4.8.

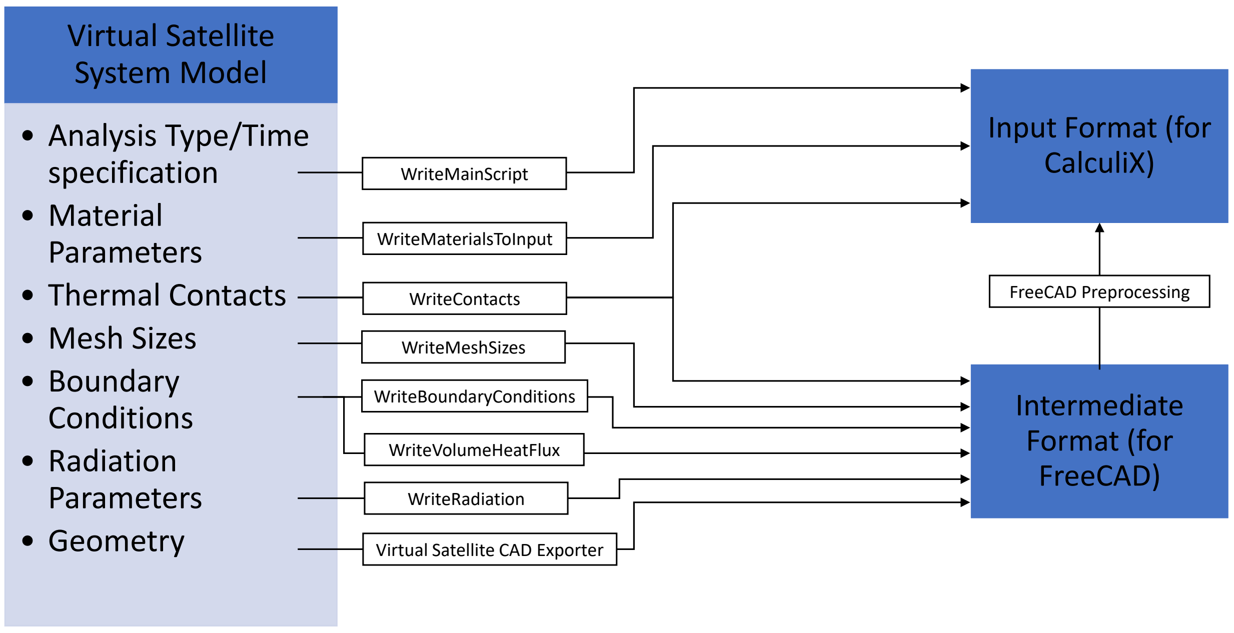


Figure 4.8: Overview of all model parts exported from VirSat for thermal simulation. The names in the boxes represent the name of the corresponding app that exports the aspect. The arrows point to the target format of the exported data. Source: Own representation.

4.2.3.3 Contacts Export

The *WriteContacts* app gathers information about thermal contacts modeled in VirSat. For each contact to be fully defined in CalculiX, the three mandatory attributes are comprised by the two involved components and the gap conductivity characterizing the contact. Consequently, for each contact at least these three attributes must be extracted from the data model and written into a readable file. Since the model of contacts in CalculiX also requires the specific surface numbers of the involved component faces, not all input needed for the contact model can be supplied by this app. As FreeCAD is the first instance in the workflow that considers single surfaces of the geometries, some of the actual input data has to be supplied by FreeCAD at a later stage of the workflow.

However, this app writes the ”add_contact.inp” file which defines the name of each contact (”SI1” counting up) and the gap conductivities. This file is already in the input format, but not sufficient for specifying the contacts. Subsequently, FreeCAD writes the missing information in a separate file using the contact name to map the parameters right (see section 4.3.2). To make sure that each contact name actually refers to the right contact in both programs, two contact validation files are created in VirSat. Each line of both files contains the names of the master and slave contact partners. The line itself represents the contact number. The app in VirSat writes the identifier of the two contact partners into the very first line, which is line number 0. The FreeCAD script can then open the files and compare the two identifiers with the detected contacts. This way not only the right order of the contacts is ensured (to match the already created ”add_contact.inp”

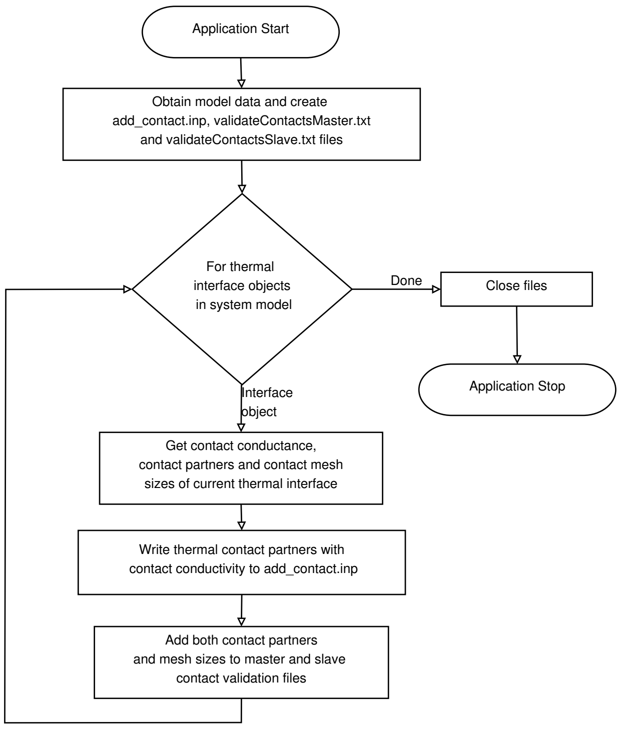

Application Start
Obtain model data and create add_contact.inp, validateContactsMaster.txt and validateContactsSlave.txt files
For thermal interface objects in system model
Done
Close files
Application Stop
Interface object
Get contact conductance, contact partners and contact mesh sizes of current thermal interface
Write thermal contact partners with contact conductivity to add_contact.inp
Add both contact partners and mesh sizes to master and slave contact validation files

Figure 4.9: Flowchart of the app that exports the thermal interfaces to the input file as well as to master and slave contact files. Source: Own representation.

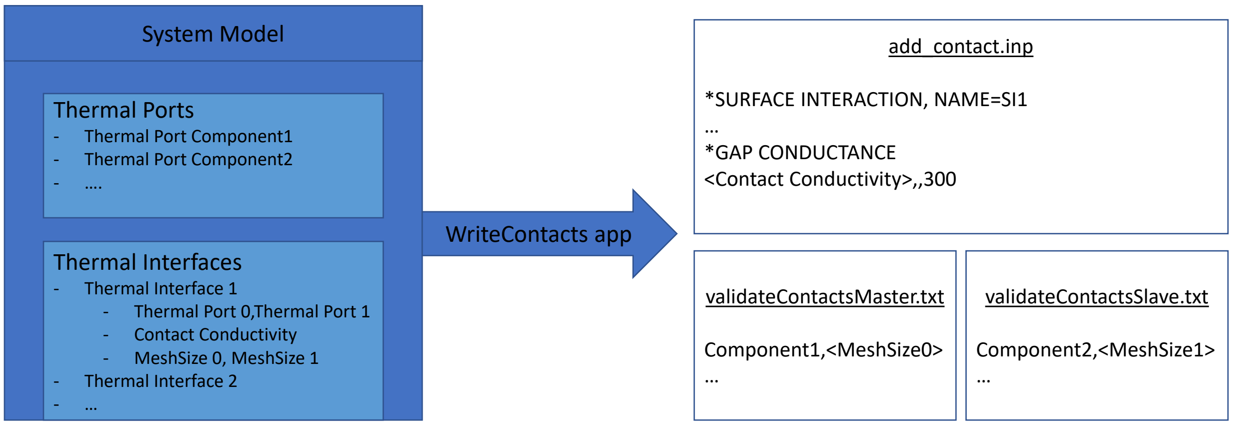


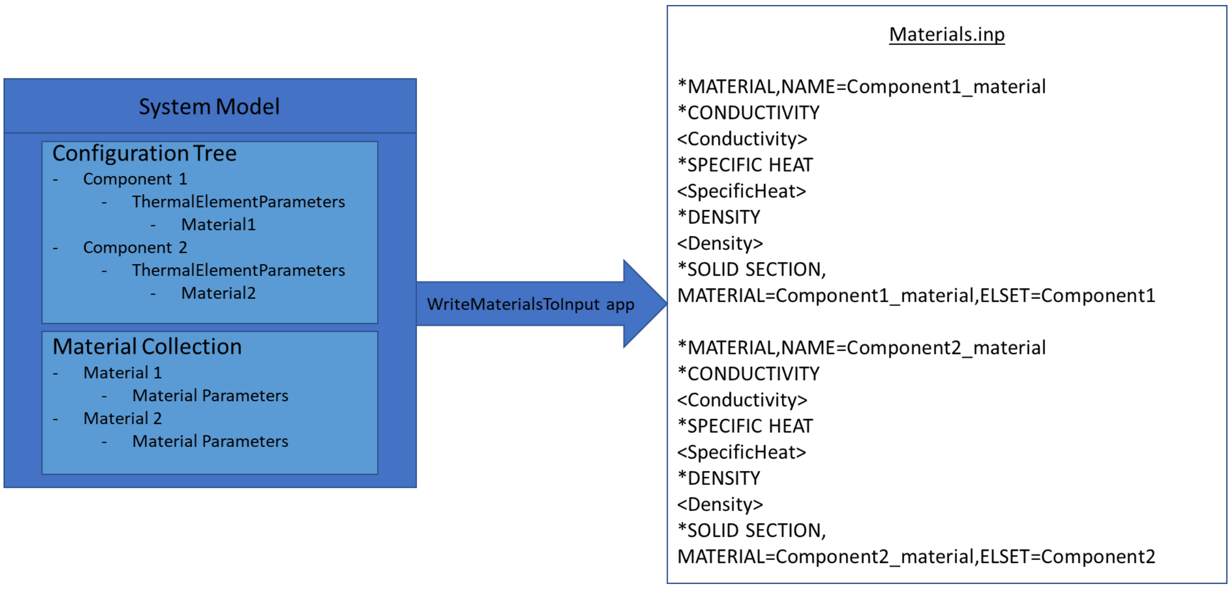
Figure 4.10: Parts of the system model the "WriteContacts" app accesses and the format of the three created files. The two *.txt files are passed to FreeCAD, while the *.inp file is directly used in CalculiX input. Source: Own representation.

file), but also the contacts detected by FreeCAD are validated and the integrity of the data for all contacts being fed into the simulation is guaranteed. This is because detected contacts without data from VirSat as well as pure data without the appropriate contact detected will not be considered in CalculiX. From the flow diagram of the app, depicted in Fig. 4.9, its logic can be obtained. Basically, the app first accesses the relevant model data and creates the three already mentioned empty files. Then it iterates through all thermal interfaces modeled in VirSat and retrieves the two ports' parent component identifier and the thermal gap conductivity. Afterwards, those values are written to the corresponding file.

In addition to the three mandatory attributes of each contact, the possibility to specify local mesh sizes at the contact location is offered for both sides of the contact. They are exported by writing the specific mesh size for each contact side to the master and slave contact validation files. In each line, right after the the name of the corresponding component, the characteristic mesh size for this contact is written. In the FreeCAD script the numbers are then obtained from these files and the relevant parts of the mesh are sized accordingly. Fig. 4.10 shows the input and output of the app, where the lower two files are for further processing in FreeCAD and the upper file is one of the two input files for contact specification in CalculiX, the other one being provided by the FreeCAD script.

4.2.3.4 Material Export

Writing the material data to the input is done by the *WriteMaterialsToInput* app. Fig. 4.11 shows the parts of the system model this app accesses, as well as the output file's structure, that is generated. To actually use a material in CalculiX, it has to be assigned to a set of elements. As there are no element sets at this stage of the workflow, since they are created after meshing, instead the identifier of the element is used as the element set name. This is shown in Fig. 4.11, where the corresponding component name is called as the element set (*ELSET*) to assign the material to, after each material's parameters are defined. To keep the input consistent, the corresponding element set must be created. It is done automatically in FreeCAD, as described in section 4.3.2.3, so that there is no risk of missing to create it, ensuring the consistency of the input model. The program flow starts with obtaining the relevant model data, meaning it gathers all structural element instances in a list. Afterwards, the blank text file "Materials.inp" is generated. Then, the app iterates through the list, checking for a visualization and the availability of the thermal element parameters defined in section 4.2.2.5 If either one of the two is not available, iteration continues as it is assumed that the current element instance is no physical component. If the parameters are available, relevant material data is written into the file. Relevant data in this case is comprised of material name, thermal conductivity, specific heat, density, as well as the name of the component the material shall be assigned to. These elements are needed to fully specify materials in CalculiX.



```
*MATERIAL,NAME=Component1_material
*CONDUCTIVITY
<Conductivity>
*SPECIFIC HEAT
<SpecificHeat>
*DENSITY
<Density>
*SOLID SECTION,
MATERIAL=Component1_material,ELSET=Component1

*MATERIAL,NAME=Component2_material
*CONDUCTIVITY
<Conductivity>
*SPECIFIC HEAT
<SpecificHeat>
*DENSITY
<Density>
*SOLID SECTION,
MATERIAL=Component2_material,ELSET=Component2
```

Figure 4.11: Conversion of system model data to input file text that is done by the "WriteMaterialsToInput" app. Source: Own representation.

4.2.3.5 Radiative Parameter Export

For exporting radiation parameters and correctly arrange them in the input files, the *WriteRadiationInput* App was created. The format for feeding these parameters to CalculiX is somewhat complicated to obtain since every single mesh face must have its emissivity assigned separately. The identifier for the face consists of the number of the element the face belongs to and the number that very face has within this element. Taking into account that VirSat does not interact with the mesh at all, this means that an intermediate step has to be introduced in order to enable the correct allocation of the emissivities in FreeCAD. For this app it is important to consider that it is possible to assign radiation parameters to individual faces, as well as to complete components.

All of these values must be interpreted correctly by FreeCAD. For this, the first line always contains the overall emissivity and absorptivity of the component. After writing this line, the following lines are optional. What happens is that the app obtains all single face emissivities that are present within this components thermal data. If one is found, the face number is written to the file. Separated by a comma, the actual emissivity and absorptivity values for this face are added to the same line. The structure of the file is depicted in Fig. 4.12. If there is no single face parameter for this component at all, it consists of one line only. This way, the file can be opened in FreeCAD, first assigning the overall radiation parameters to the complete component, then assigning the individual ones by finding the corresponding mesh faces that belong to the specified part face. Thereby, the before assigned general value is simply overridden by the face specific one.

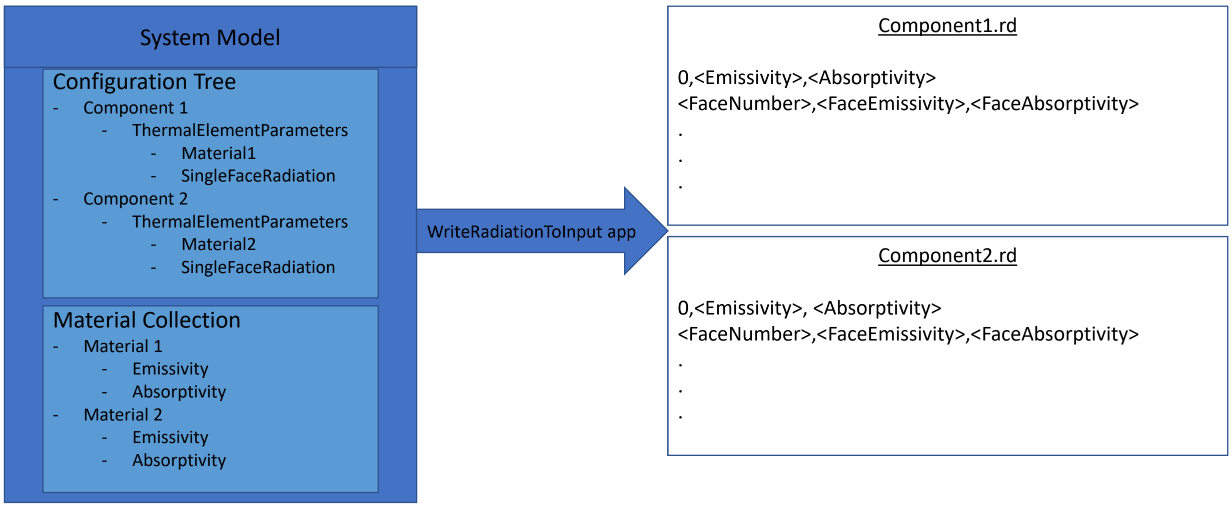


Figure 4.12: Conversion of system model data to the intermediate ".rd" file that is done by the *WriteRadiationToInput* app. The first line of the file represents the overall component radiation properties, while the following lines are optional and specify emissivities and absorptivities for specific faces. The values in brackets represent the values deposited in the system model. Source: Own representation.

4.2.3.6 Volumetric Heat Flux Export

WriteVolumeHeatFluxToInput is the app to take account for the heat dissipated by a component heating itself due to electrical processes, or even cooling itself to model a cooling effect by assigning negative heat fluxes. The flow is similar to the apps before. First, all structural element instances are collected, then the ones with thermal element parameters are accessed and the value of the thermal power balance is obtained. It is then written to a *.bfl text file that is named after the name and identifier of the component. Since the body flux is given to CalculiX as heat flow per unit volume, the absolute value only is not enough to reasonably specify the body flux; also the volume of the body is necessary to compute the volumetric heat flow. Although the volume of a component is technically present in the VirSat visualization, it cannot be accessed easily. It would be possible to calculate it from the shape of the geometry and the parameters, however due to the amount of different shapes it would also be unnecessarily complex to calculate the volume for all possible shapes. Consequently, only the absolute value of the heat flow is written to the file and then converted in FreeCAD, which supplies the value of the volume more conveniently, to a volumetric heat flow. Another benefit of this is that the volume of the actual meshed component can be used instead of the real volume. This way the absolute heat flow in the analysis will have the value specified in VirSat, while otherwise it would slightly deviate due to the volume difference between ideal body and mesh.

4.2.3.7 Further Boundary Condition Export

The three boundary conditions left are exported with the *WriteBoundaryConditions* app. Since the volumetric heat flux and radiation are already covered, what is left are face heat fluxes, face temperature and body temperature. These three boundary conditions are all stored in the ”BoundaryConditions” structural element and are all accessed with one app.

As some apps before, this app iterates through all element configurations and determines if they are components by checking if a visualization and thermal parameters exist in it. If so, two files named after the name and identifier of the component are created. One is for both temperature boundary conditions, the other one for face heat fluxes. Then, an iteration through all elements within the ”BoundaryConditions” element obtains all boundary conditions to be set. For each one, it is determined what kind of boundary condition it is. Then it is appended to the according file. The body temperature boundary condition is just written as as a number into a single line, FreeCAD then later recognizes the single number as a volume boundary condition and identifies all elements of the body to set the temperature. For face boundary conditions, the face number and the value of the condition is specified. Then, in FreeCAD the according mesh elements located in the face are identified and the value of the boundary condition is assigned to the right mesh element faces. In the temperature boundary file, FreeCAD can distinguish between face and body boundary by determining the amount of values separated by commas in a line. In addition, if there is a body condition, a face condition will overwrite the body condition for this specific face as it will be applied after the body condition. This way, it is technically possible to define a body condition with single faces having different temperatures.

4.2.3.8 Mesh Size Export

The export of the individual mesh sizes is done with the *MeshSizeExport* app. It creates the ”meshSizes.txt” file and writes the mesh size for each component together with the name and identifier to it. The mesh sizes are obtained by accessing the ”MeshSizes” element, where the sizes are stored. The way the app works is that it iterates through all element configurations to find all components. Then, for each component it iterates through all specified mesh sizes and identifies if the name of the component belonging to a mesh size matches the name of the component of the current iteration. If so, the mesh size is obtained and written to the text file, among the component name and identifier. If there is no mesh size specified for a component, the value 0 is used instead, resulting in Gmsh automatically determining the size of the mesh elements.

4.2.3.9 General Setup Data Export

To complete the export apps, the ”main_input.inp” is generated. It is done by executing the *WriteMainScript* app. It can be imagined as the trunk of a tree to which every branch, representing all other input files that were generated, is connected. The task of the main file is to set the sequence of simulation setup and as a part of this, invoke the single input files in the correct and complete order. Thus, it is important to make sure that all relevant input files are invoked and the simulation setup is complete. Another task is to specify some general information regarding the simulation that only needs to be stated once.

Consequently, this app is the most complex one and a flow diagram is supplied in Fig. 4.13. First, the ”ThermalAnalysis” structural element containing the general simulation setup information is retrieved. Then, the empty ”main.inp” file is created. Since geometry is the first information provided in a CalculiX input file, the first lines in the main file invoke the mesh geometry files of every component, which are created by FreeCAD as described in chapter 4.3.2. As they contain all nodes, elements and sets, they are a necessary foundation for specifying loads and properties. Because the geometry files are not necessarily already existing when the main file is generated, the naming of the files follows a convention. The component name and identifier is used as file name. An invocation for every structural instance that satisfies the conditions of having a ”ThermalElementParameters” element as well as a visualization with a shape assigned is created. This way, errors resulting from incomplete elements in VirSat are avoided.

After the geometries, a fixed line for physical constants is generated followed by setting the initial temperature. The initial temperature is first set to 300 K for all nodes. This way it is ensured, that all nodes have at least one initial temperature. For steady state simulations the initial temperature does not have influence and the default value is used. But if the analysis type is transient, the app gets the initial temperature of every component from the thermal element parameters using the same conditions as used for the geometry file invocation. The initial temperatures are then written to the input assuming there is a node set existing for every component with the component's name at the very time this line is executed. Next up is the material and contact properties. The material input file is already completed as it contains material data and the corresponding element set names. If the element sets are defined before, which they are, the materials are assigned to the correct element sets. The contact definition consists of two files which are called after each other. The first file is the one created by the *WriteContacts* app, the second file ”contact_surfaces” is created by FreeCAD.

Afterwards, the *Step* keyword is written starting the part of the script where loads and boundaries are created. Now, when it comes to the main file, there are differences in this part's structure depending on whether it is a dynamic or steady state simulation. Dynamic simulations need some extra input information such as the time parameters. So, first, the analysis type is specified by either using the *Steady State* parameter after the *Heat Transfer* keyword or no parameter at all, indicating a dynamic simulation.

For the dynamic simulation case, it is necessary to hand over the time parameters in the line below. This includes start time, which is always set to 0, stop time and time step, all separated by commas.

Then, all heat flux and radiation files are invoked, again using the same condition that was described earlier for the geometry file invocation to ensure only the files for existing and completely defined components are invoked.

For post-processing, the standard settings are set, such that an output file is generated, that hosts temperatures of the nodes, as well as heat fluxes between the elements.

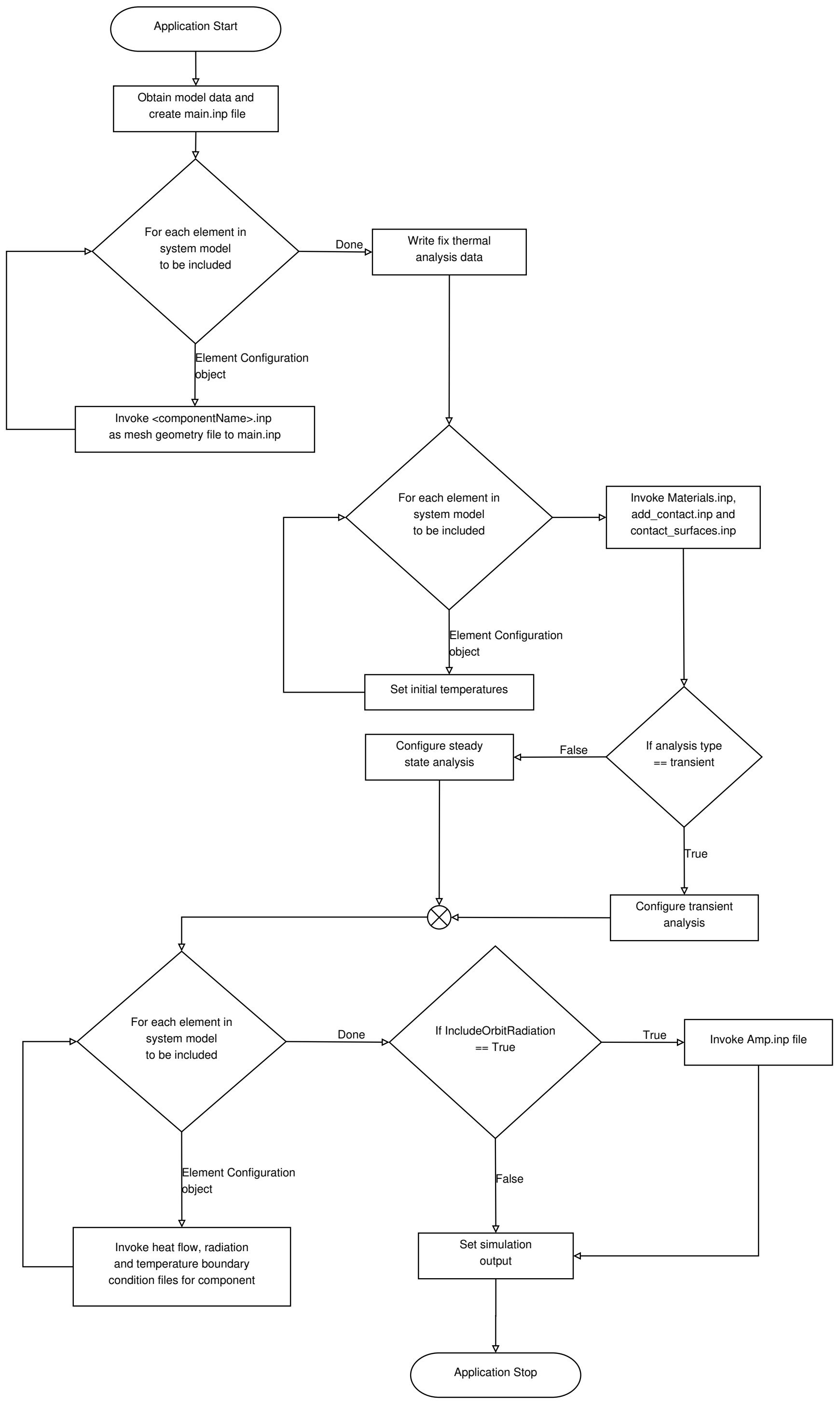

Application Start
Obtain model data and create main.inp file
For each element in system model to be included
Done
Write fix thermal analysis data
Element Configuration object
Invoke <componentName>.inp as mesh geometry file to main.inp
For each element in system model to be included
Invoke Materials.inp, add_contact.inp and contact_surfaces.inp
Element Configuration object
Set initial temperatures
Configure steady state analysis
False
If analysis type == transient
True
Configure transient analysis
For each element in system model to be included
Done
If IncludeOrbitRadiation == True
True
Invoke Amp.inp file
Element Configuration object
False
Invoke heat flow, radiation and temperature boundary condition files for component
Set simulation output
Application Stop

Figure 4.13: Flowchart of the app that exports the thermal model to the main input file that invokes the other input files. Source: Own representation.

4.3 Processing with FreeCAD

Since VirSat is a tool for modeling a satellite and not for handling finite element data or detailed geometric data, this needs to be outsourced to a different software. For this, the CAD export function is a very useful one. It not only allows to change geometric properties in an external CAD software and feed them back to VirSat, but it also enable to further process it to enable conducting simulations with the geometry supplied by VirSat. Processing the geometry by generating a mesh that matches it and supplying this mesh to CalculiX is the central task FreeCAD was intended for in this workflow. However, additional pre-processing steps have to be executed. This comprises the allocation of the boundary conditions for example. To further smoothen the workflow an automated function for this is desired, saving the effort for manually doing these tasks. Having in mind that the workflow works best, the less complex it is and the fewer different software packages are included, FreeCAD's Python console was considered for this and turned out as well-suited for most tasks. It offers direct access to all geometric model properties and many mesh properties. In addition the Python programming language offers basic functionality needed for automation of these steps like opening files, writing lines and performing calculations. Thus, FreeCAD's task, originally intended to only process the geometry was extended by essentially all other necessary pre-processing steps. To automate certain tasks, FreeCAD's functionality includes storing macros for scripting, that are comprised of Python code to be executed. Since numerous tasks are executed using a script that was developed in the course of this work, this script is one of the key components in this work. Thus, it is introduced and explained in this section. A list of the single functions the FreeCAD script covers is provided in Tab. 4.1. A top level flow diagram of the FreeCAD script sequencing is provided in Fig. 4.14. In the course of this section, first all pre-processing regarding the geometry is presented. This covers splitting contacting faces into the actual contact face and the residual contact-free face, meshing, finding and processing the contacts between bodies, and finally refining certain parts of the mesh. Secondly, those boundary conditions and surface properties that VirSat was not able to set, are processed and set. The section closes with the invocation of external factors as solar radiation.

Table 4.1: FreeCAD script functions

Function name	Function content
findAllContacts()	Find contacts and add the contact information to a globally available matrix
createMeshAndGroupsAndInputFile()	Create the mesh with Gmsh, then create the mesh subsets for assigning all boundary conditions and properties, finally write everything into the mesh file
makeInputConsecutive()	Open mesh file and change node/element numbers to make the numbers consecutive across all components

addNodesetForObject()	Add all nodes of a component to a new mesh group
addElementsetForObject()	Add all elements of a component to a new mesh group
setFaceEmissivities()	Read VirSat radiation output file, then assign the correct emissivity value to all faces of a component and write the corresponding input file for CalculiX
applyHeatFluxBoundaryConditions()	Read VirSat (face) heat flux file and apply the heat fluxes to the according element faces, then write the input file for CalculiX
applyVolumeFlux()	Read VirSat (volume) heat flow, divide it by the volume of the corresponding component's mesh and write the resulting volumetric heat flow to the input file for CalculiX
applyTemperatureBoundaryConditions()	Read VirSat temperature boundary condition file (volume and face) and apply the temperatures to the corresponding nodes, then write the input file for CalculiX
validateContact()	Compare a detected contact with the specified contacts from VirSat, if match is found return "True"
writeContactToInput()	Write the matrix of validated contacts to the input file for CalculiX
makeContactFaces()	Pre-process the geometry before the actual script execution by creating new bodies with separate faces at contact areas that are used instead of the original ones. In addition, read the contact file from VirSat and cut the corresponding "Slave" component if an overlap is detected
reset()	Reset the geometries to the initial state (Delete all types of bodies except the type the imported ones of VirSat are)
writeAmplitudeFile()	Process the Sun and Earth orbit files to a sequence of loads for each mesh face
findMeshFaceNormal()	Determine the normal vector of the face formed by the three nodes handed to this function using the cross product
findMeshFaceOrientation()	Determine the orientation of the normal vector from the order of the nodes

getEarthData()	Read the CSV files and write the Earth vector and albedo reflection angle for each timestep to a list
getSunData()	Read the CSV files and write the Sun vector and Sun intensity for each timestep to a list
getCosineOfAngleBetweenVectors()	Determines the cosine of the angle between two vectors
determineObstruction()	Determine if the path from the first node of a face to another point is obstructed by another part of the geometry

4.3.1 Pre-processing the Geometry

Ignoring the part hierarchy information for now and only considering the geometry, the exported model consists of individual bodies representing each component. These separated bodies are not related each other. However, if a face of one body has contact to another face, a part of this face is obstructed by the body it has contact with. Thus, on a logical level this face is separated into two faces, but in the geometric model there is still only one face. If one now assigns a load to that face, representing for example solar radiation, it might be desired that this load only covers the unobstructed part of the face, as it would be in reality. But, as mentioned, there is only one face in the geometrical model. So it is not yet possible to assign the two parts of the faces individually.

Another benefit underlining the usefulness of splitting up the faces is refining the mesh. Refining the mesh at contact areas in particular, not on the complete face of the contact, would improve the accuracy of the simulation as it is proved in chapter 5. Having these two arguments in mind, a solution for splitting up a face with a contact into the actual contact area and the free area seemed to be worth striving for. Certainly, the possibility of a quick solution was investigated, however due to the way of FreeCAD handling and storing faces and bodies, there was no apparent solution to this. Using boolean functions with bodies that have faces which are only lying on each other and not actually intersecting do not return anything. The only predefined function that provides an overlap between the faces is the *section*() function. It returns a set of edges describing the outer wire of the contact face. All attempts to add this wire as a face to the bodies were unsuccessful. In the end, a hint from one of the FreeCAD core developers enabled the use of a workaround.

This workaround is implemented in the *makeContactFaces*() function in the FreeCAD script. The way it works is that a list of all components' geometry objects is created. Out of this list, a so called boolean fragment is generated. If the bodies forming the boolean fragment are overlapping, the overlapping area is cut out of the two bodies forming three new solids, two of them having the shape of the original ones just with the overlapping area removed and the third one has the shape of the overlap. If only the faces are in contact, only two new solids are created,

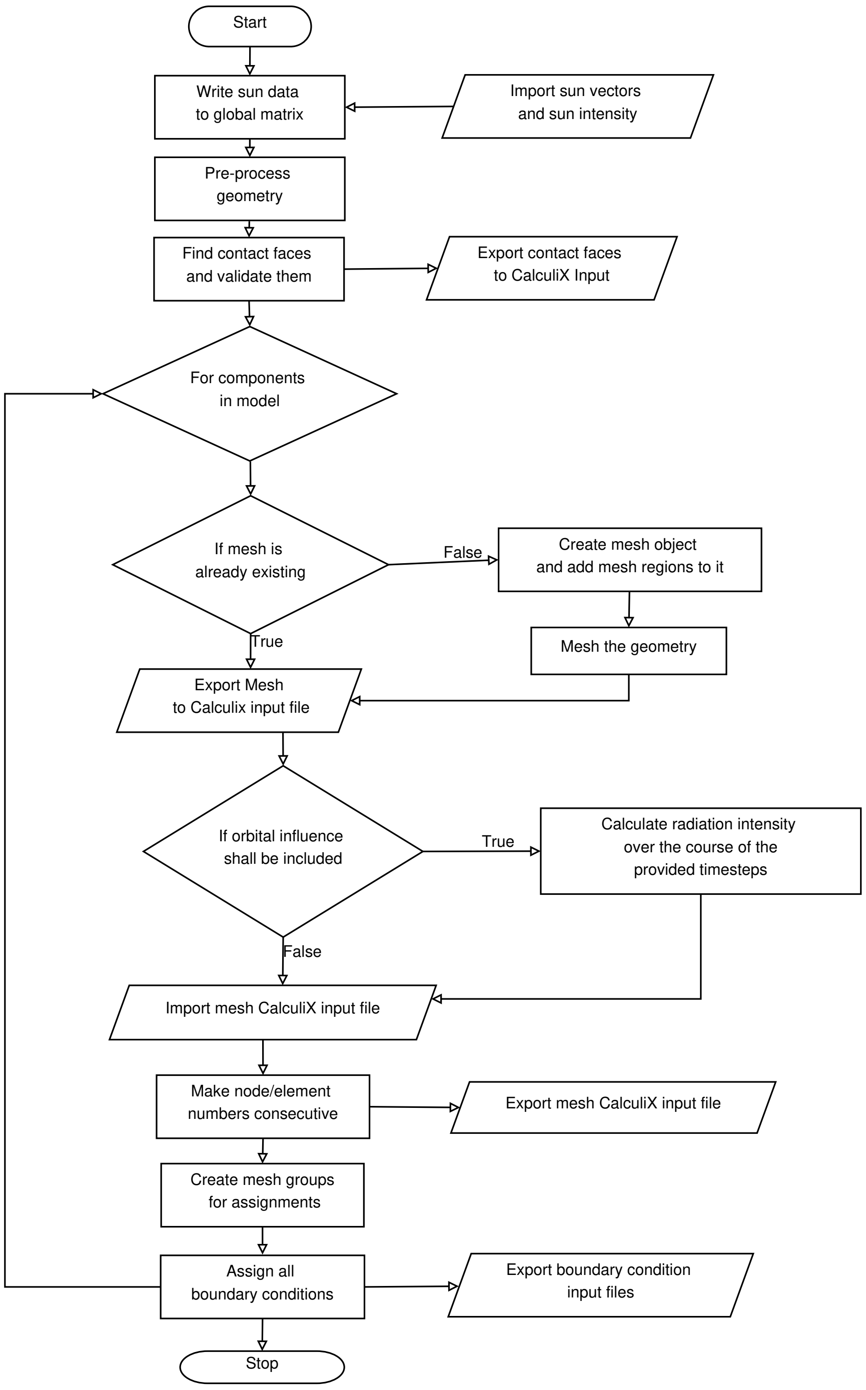

Start
Write sun data to global matrix
Import sun vectors and sun intensity
Pre-process geometry
Find contact faces and validate them
Export contact faces to CalculiX Input
For components in model
If mesh is already existing
False
Create mesh object and add mesh regions to it
True
Mesh the geometry
Export Mesh to Calculix input file
If orbital influence shall be included
True
Calculate radiation intensity over the course of the provided timesteps
False
Import mesh CalculiX input file
Make node/element numbers consecutive
Export mesh CalculiX input file
Create mesh groups for assignments
Assign all boundary conditions
Export boundary condition input files
Stop

Figure 4.14: Flowchart of the top level functions the FreeCAD script performs. Source: Own representation.

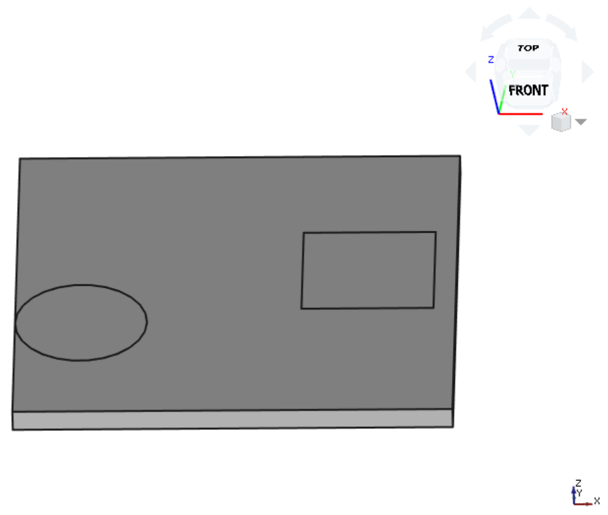


Figure 4.15: Baseplate with additional face at cylindrical contact area on the left as it is created by the geometry pre-processing. Source: Own representation.

each one with the contact area as an individual face. Now, a solid is just a subset of a proper geometry object, meaning it has to be converted to the latter. This is done with the boolean function *common*() that gets the intersection between two shapes as a real part object that can be used for further processing. One of the two shapes being the whole shape of the boolean fragment and the other one being the one of the original objects. What is returned is a new object with the original shape and the desired single faces for each contact as depicted in Fig. 4.15. With this pre-processing executed, it is important to keep in mind that the number of faces as well as the assignment of the numbers to the faces is changed. So, the face numbers of the original object differ from that of the new one. This is especially important when defining properties on specific faces in the thermal concept in VirSat as the new face numbers are used in FreeCAD to assign the defined properties. Thus, if a property like a boundary condition or a single face radiation parameter is assigned in VirSat, this pre-processing step has to be done first, to know which face to assign the property to.

4.3.2 Automated Meshing and Mesh Groups

To conduct a finite element analysis of a system one needs to convert the geometry into a discrete set of elements and nodes. This is done by creating a mesh. A mesh consists of elements which again consist of interconnected nodes, single points in the coordinate frame. For highly accurate simulations, one needs very fine meshes that are refined manually and thereby optimized for the type of simulation they are intended for. This task is usually done by experienced thermal engineers and requires experience in thermal analysis. Still, new automated meshing algorithms are continuously developed and mesh quality is improved with better algorithms.

When executing the automated FreeCAD script, the focus of the simulation is not set on obtaining the highest accuracy thermal analyses with highly refined meshes but more on obtaining relatively quick simulation results as early as possible in the project life cycle. It is therefore convenient that the mesh is created as fast as possible while the mesh quality is still satisfying for the demand. To ensure this, the mesh element size can be set for the individual components in the thermal concept in VirSat. Since in the early design phases reasonable effort is the most important aspect in conducting thermal analyses, the mesh is generated automatically in this automated part of the workflow. However, the option to do more sophisticated analyses with customized meshes shall be enabled in this workflow as well and is addressed in section 4.3.3.

As it was described before, the geometry from VirSat is exported using the *Cad Export Wizard* function. The export format is JavaScript Object Notation (JSON), which is not one of the usually used CAD formats. The reason this format is used is that it can display the hierarchy relations between single parts and subsystems. This is necessary, because the original intention of the export function was to export the project to FreeCAD to change geometric features and then reimport this updated version in VirSat. To keep the project consistent, the hierarchy information had to be included in the export model. Since the JSON format is not a usual CAD format, CAD software packages cannot handle it. In FreeCAD however, DLR-SC developed a Virtual Satellite Workbench enabling the import of the geometry in the JSON format with the correct project hierarchy. Thus, FreeCAD is the only CAD software that is really able to process the geometry from VirSat. Because FreeCAD also has a FEM workbench, allowing to create mainly mechanical finite element analyses, some infrastructure in the area of finite element analyses is already present. For example, there are two mesh tools that can be directly executed by FreeCAD; Netgen and Gmsh. The latter one, offering more customizability of the mesh, is chosen to be used here.

First, to test the FEM capability of FreeCAD and the mesh tools, meshes were created using the graphical user interface. However, scripting the mesh creation is much faster and therefore exclusively the process of creating a mesh with a Python script is of relevance here. The same also holds for several other FEM functions used in the workflow.

In the developed FreeCAD script, a main loop iterates through all individual components, essentially executing all of the functions following for each component one by one.

4.3.2.1 Mesh Generation

Generating a generic mesh with the Python console in FreeCAD is done within a few lines of code, as seen in listing 4.3. The first thing to do is to create an empty FEM mesh object by importing the FEM workbench and using the makeMeshGmsh() command handing the name of the object as a parameter. Then, the part that is supposed to be meshed and some more parameters, if desired, are assigned to the empty mesh object. After that, the mesh object is specified to be meshed using Gmsh. It is also possible to define specific areas within the part, so called ”mesh regions”, which are meshed with different specific length parameters for example. An example of this is depicted in Fig. 4.16, where the face on the lower side of the cube was defined as a mesh

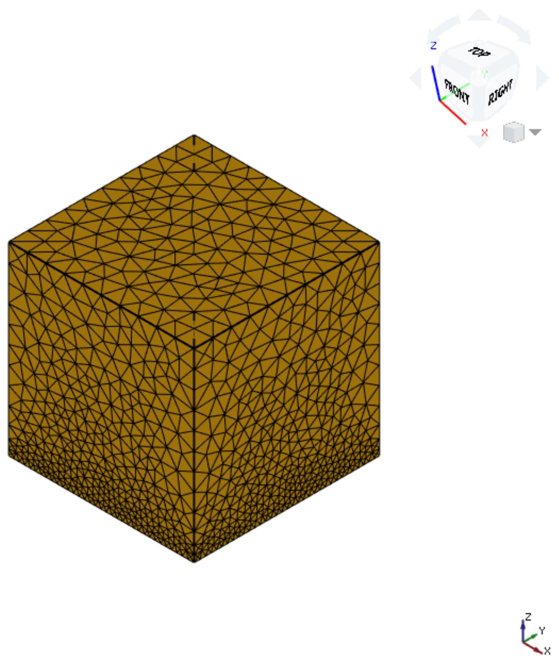


Figure 4.16: Meshed cube geometry, where the face on the bottom was assigned as mesh region with a lower element size before meshing was executed. Source: Own representation.

Listing 4.3: Example for creating a generic mesh of a part using the Python console in FreeCAD

```
import FreeCAD, ObjectsFem
from femmesh.gmshtools import GmshTools as gt

#Creation of the new empty mesh object
femmesh_object = ObjectsFem.makeMeshGmsh(App.ActiveDocument,"MeshExample")
#Assignment of the Part ("Box") to be meshed to the mesh object
femmesh_object.Part = App.ActiveDocument.Box
#GMSH meshing
gmsh_mesh = gt(femmesh_object)
gmsh_mesh.create_mesh()
```

region with a lower maximum specific mesh length. This function is used for refining the mesh on contact areas, as described in section 4.3.6.

Another concept that is used extensively are mesh groups. A mesh group is a collection of nodes, elements, faces or lines that are grouped to assign loads to the whole collection instead of single instances. This improves the readability of the input files and already enables the allocation of loads in files created by VirSat apps. They can just assign the parameters to a mesh group name that is not further specified at that point of the workflow. Consequently, this mesh group must be actually initialized in FreeCAD, otherwise there will be an input error in CalculiX since there are values allocated to a mesh group that does not exist. So, it is important to ensure that the set of mesh groups is consistent when the input is handed over to CalculiX.

To generate the mesh, three things are needed in this workflow. First, the actual geometric bodies to create the mesh of. They are provided by the export function in VirSat. Second, the general mesh size of the bodies. For this, a small function is executed before the actual meshing starts. This function reads the ”meshSizes.txt” file that was exported by VirSat and creates a list with all component’s mesh sizes. Third, the contact regions and their corresponding mesh sizes. The contact regions are obtained by first pre-processing the geometry to isolate single faces for the contact areas as it is described in section 4.3.1, and then finding the actual faces where the contacts are as described in 4.3.4. The latter also supplies the sizes to be applied for the contact regions. With these three aspects the mesh is generated by Gmsh.

4.3.2.2 Consecutive Meshes

Now, since every component is meshed individually, for every component the default node and element numbering consequently starts at 1. As all meshes are later fed to the same CalculiX input, the same node and element numbers are actually defined more than once throughout the whole input if there is more than one meshed component. This causes the problem that the input deck is not consistent and various errors are the consequence. Because it is not possible to influence the node and element number that Gmsh distributes, another solution was to be considered. The first approach was to change the node number within FreeCAD after mesh generation. As it is possible to access most of the parameters of the geometry and change them, accessing the mesh parameters like this was the most obvious approach. However, after FreeCAD imports the mesh, it does not allow to change the nodes and elements in any way.

As a consequence, a different solution had to be chosen. That is, accessing the exported ”*.inp” files that contain the mesh information and manipulating the numbers inside. The mesh files are text based, which makes it possible to read and change it. However, it is required to cycle through all lines, remove the artifacts of the text file to convert the entries of each line to a list of actual numbers to work with. Furthermore, the correct lines need to be identified to change the right values. To execute all this, the function *makeInputConsecutive*() was created. It takes the number of already given nodes, the number of nodes for the current component, the same for the elements, and the path where the mesh files are as arguments. First, it is determined in which ranges of the file nodes, elements and mesh groups are defined. This follows a fixed relation and can be obtained with the given parameters. Then, the function iterates through all lines of the file and determines in which of the three areas it is.

For the nodes, the function obtains the first entry of the line, which is the node number, and changes it accordingly. If the line index is in the range of the element definition, analogously to the nodes the first number in the line, the element number is replaced with the number of already assigned elements plus another running variable, counting the number of assigned elements in that file.
In addition to this, in each line there are ten more entries representing the node numbers of the nodes the element consists of. Consequently, these have to be changed as well, which is why there is another loop iterating through all entries of the line except the first one, changing the numbers to the old number plus the fixed number of nodes already assigned in the previous components.

If the line index exceeds the upper border of the element definition area, that essentially means it is in the mesh group area. Since all mesh groups are created after this step, no additional manipulation is necessary here. But it is important to mention, that if mesh groups would have been created at this point, the nodes or elements they contain will have to be changed accordingly as well.

4.3.2.3 Mesh Group Generation

After executing this function, the mesh files are updated and the node and element numbers are consistent throughout all components. The next step is to create all the mesh groups that are used for assigning values to the geometry by identifying the desired nodes or elements and adding them to a mesh group with a certain name. It was mentioned before, that the export apps in VirSat rely on certain mesh groups to be existent. These mesh groups of course have to be created. Since the material export app assigns the materials to an element mesh group with the name of the component, for each component a mesh group with all of the component's elements is created. Analogously, temperature or volumetric heat flux boundary conditions can be applied to a whole component, also using these mesh groups. Since these conditions are always applied to nodes and not to elements, a node set of the same name as the element set is generated which in fact just extends the before created element set by the selected nodes so that the set inherits both, elements and nodes.

For every contact that was detected by the function described in section 4.3.4, a surface mesh group for the two involved faces is created. This is done because CalculiX needs these faces to be specified when the contacts are defined. The face mesh groups are named consistently after their component's label first and the component label they are touching second. The expression "Master_Contact_" and "Slave_Contact_" is also used since CalculiX uses similar expressions. There is no specific routine assigning master and slave to the faces since for thermal simulations, it makes no difference which face is which [42]. The mesh group is created within the mesh file of each component. By obtaining all mesh faces that comprise the whole actual face of the component, which is in contact with another face, then writing the mesh element number of each mesh face and the number of that mesh face within the mesh element into the mesh group, the contact face is defined well enough for CalculiX.

With the contact faces defined, the mesh creation and subsequent addition of mesh groups is completed and the mesh itself is fully prepared for thermal analysis. With the material being assigned to the created mesh groups, only boundary conditions are left to be assigned.

4.3.3 Manual Mesh Generation

In section 4.3.2 it was discussed that the automated meshing is part of the workflow to be executed when a fast simulation result is desired. However, to keep this workflow variable and usable also for simulations in later design phases, when more time is spent crafting and refining meshes manually, a corresponding feature was added to the FreeCAD script. With this feature, it is possible to create a mesh or import a created mesh of one or more components before executing the script. This mesh is then recognized by the script, provided that it follows the naming convention. It has to be named exactly like the component, with a ”_001” appended. This is also the standard way FreeCAD renames the mesh element, when it is created with the same name as the component itself. When a mesh is recognized, the script will not create a new mesh for that component and use the existing mesh for further pre-processing (starting from mesh group generation). This way, parts of the model, or even the whole model, can be meshed to individual needs, while still using most parts of the automated workflow, significantly easing many parts of simulation pre- and post-processing. At this point it is only possible to use tetrahedral meshes if it is desired to include solar radiation, because the order of nodes within an element that form specific face numbers is specific to the tetrahedral mesh. However, other mesh types could be added with relatively low effort, only specifying the order of nodes for other mesh types.

4.3.4 Finding Contacts between Components

To specify the contact faces and the relevant mesh faces for the contact face mesh group as described in the previous section, it is necessary to find them first. It is important to keep in mind that after execution of the steps described in section 4.3.1, every component in the geometric model has a dedicated face for the exact contact area with any component. Now, reliably determining these precise faces’ numbers requires some effort as there is no function for doing this directly. Thus, to compare two faces, it is necessary to look at their composition.

In FreeCAD, faces ultimately consist of single ”vertices”, that are connected by ”edges”, which are attached to each other to form a closed ”wire”. At least one wire forms a face and if a face has more than one wire, that means it is not sufficiently described by a single sequence of attached edges. To find contacting faces, a loop through all components’ faces is executed. Then, a second loop iterates through all faces of the components different to the component of the already selected face. For each face pair the following operations are executed to detect contacts. First, the *section*() command is used, which returns the intersection shape between two parts as an object with edges (and vertices).

There are two kinds of contacts that are being detected by evaluating the presence of edges by this command:

1. Face to face contact (desired) - contact that involves two faces of two different components, forming a contact area
2. Edge-like contact (undesired) - contact that involves two faces, forming a contact point or line

With each contact being represented by edges, it is necessary to determine if these edges represent an actual face to face contact or just a one-dimensional contact, as depicted in Fig. 4.17. To

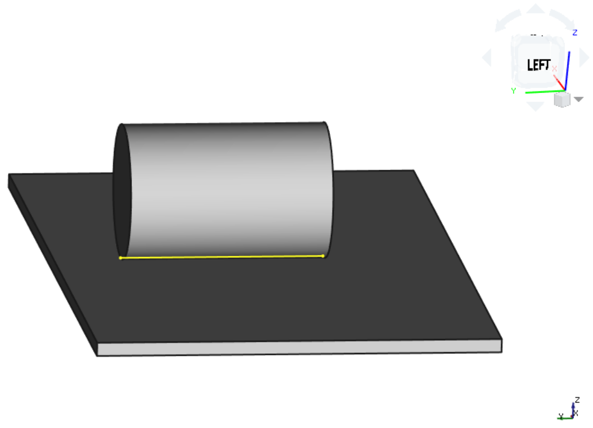


Figure 4.17: Example for a single-edge contact. The contact is only described by one edge (highlighted in yellow) and therefore no area can be assigned to it. Source: Own representation.

achieve this, the section object is undertaken a number of checks, eventually determining if a face to face contact is prevalent. First of all, there are two cases that need to be handled separately. Circular face contacts and non-circular face contacts. The special thing about circular contacts is that they consist of only one edge and therefore have to be distinguished from a simple edge contact, that is not treated as a contact here due to the lack of contact area (see section 4.3.5 for handling edge-like contacts).

Non-circular contacts are identified by first filtering only those section objects with more than one edge from the set of all possible contacts. This removes all single edge and round face contacts from the set of possible contacts. What is left are all contacts with more than one edge, thus every possible contact that is not described by a circle or single line. Now, it is possible that the contact is either generated by two faces that actually cover each other, or by at least two edges that lie on top of a surface. The latter being an undesired contact, is actually often prevalent as those contacts are regularly generated by the pre-processing function from section 4.3.1, that creates the contact area as an individual face. These contact faces are often completely enclosed by the residual of their original face, like an island. The face in contact with this newly created face is usually detected as such an undesired contact with the enclosing original face, as their two edges match at the circumference of the contact area. This case is depicted in Fig. 4.18, where the box that was removed in Fig. 4.15 was added. The face of the box that lies on the baseplate now forms the undesired contact with the face of the baseplate that encloses the contact. Many of these false contacts can be easily dismissed by counting the number of wires the faces consist of. For contacting faces the number of wires involved is always one. The enclosing faces however usually have two wires, one on the outside describing the original face and one that describes the inner border, where the contact face was cut out. However, if the cut out contact face lies at the very edge of the original face, the residual face still has only one wire that can fully describe the circumference of the face. Thus, a final evaluation has to be created to safely determine if it is a face to face contact. This is done by obtaining all vertices from the two involved faces and com-

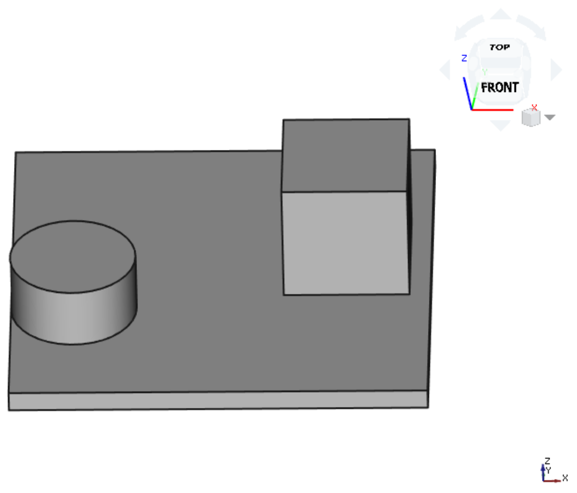


Figure 4.18: The model from Fig. 4.15 with the contact bodies added to the visualization. Source: Own representation.

paring them. If all vertices of each face have their equivalent vertex with the same coordinate in the other face, the faces are identified as equal and the contact is identified as face to face contact. If there is one vertex in one face that has no equivalent in the other face, the contact is identified as false contact.

For circular contacts the determination is different than for the non-circular contacts. A circular contact is defined by exactly one edge. Contacts with curved faces not covered with this and are handled as multi-edge contacts. This is because a curved contact face has at least one more edge than a flat contact, which automatically triggers the before discussed multi-edge contact identification.

For the flat type of contact the same applies for filtering the bordering faces of the contact as for the multi-edge contacts, with the difference that here both faces' number of edges is checked to be equal to one. If greater than one, the contact is identified as false contact, as, due to the fact that for each contact a separate face is created, a contact that is fully described by one edge can only be between two faces that are also fully described by exactly one edge. Here, the number of edges is used instead of the number of wires to ensure that the involved faces are actually circular.

If a face to face contact is detected a validation check is executed. The script opens the two files that were exported from VirSat, containing information about the contacts that are actually defined in the system model. If the contact is not found in the available files, meaning it was not defined, it is rejected and not treated as a contact in the analysis. However, if the contact between the two components was defined in the system model, then the contact pair is added to a list where all validated contacts are stored in terms of the involved components as well as the corresponding faces of the components. This list is used for the generation of the correct mesh groups of the contacts that was described earlier in 4.3.2 and for the generation of locally refined mesh areas described in section 4.3.6.

4.3.5 Point, Edge and Asymptotic Contacts

In CalculiX, a thermal contact is defined by its thermal contact conductivity [42]. As this quantity is related to the actual area of the contact, contacts without an area can only be neglected in the FEA software. This comprises point, edge, and asymptotic contacts. In reality, those kinds of contacts always have a contact area due to their imperfect shape. To enable the user to account for this, two options exist. Either, the contact must be slightly flattened manually, such that there is an actual area that FreeCAD and CalculiX detect, or this task is executed automatically. If done manually, this means the component cannot be created in VirSat any more, as such shapes are not supported by the visualization concept. Consequently, the automated version is preferred in this workflow as it is much faster and allows the geometry creation in VirSat, too.

For this, one component is slightly shifted into the other, such that the overlap is just big enough to represent the actual approximate contact area in reality. The amount of overlapping shall be determined by the thermal or mechanical engineers, who have to analyze how large the actual effective contact area, formed by the shape imperfections, is. The overlap is then chosen that way, that the resulting contact area matches the effective real contact area. The FreeCAD script geometry pre-processing (*makeContactFaces*() function and *prepareModel* script) recognizes this overlap with the boolean function *common*(). If the contact is also specified in the system model it utilizes the *cut*() function to cut out the overlapping part of one component, so that the geometry model is consistent again. In this case it is important, which component is specified first in the contact within the system model, as the first component ("Master") stays as it is and the second component ("Slave") is cut at the contact. With this, the components now have a contact area that can be processed further as a regular contact. Note that the slave components volume will inevitably change through this operation. If a volume heat flow boundary condition is applied to the slave component it will be applied to the lower volume of the component after cutting the overlap. That way, the absolute magnitude of the heat flow stays the same, but the volumetric value will slightly increase. However, since the overlaps are very small, the change in the volumetric heat flow is also marginal. Still, if a high accuracy is desired, this has to be considered.

4.3.6 Automated Mesh Refining

In every thermal model, there are certain critical areas that should be examined in more detail. Such areas could be areas of high gradients in material properties for example. Generating a finer mesh in these areas contributes to the quality of the analysis results since critical parts are calculated with more accuracy due to the higher number of elements involved. Contact areas are often this kind of critical areas, since two components with often differing material properties are coming together at these locations. Thus, it is beneficial to refine the mesh at the contact areas. This is implemented in the Python script by adding so called "Mesh Regions" to the mesh object in FreeCAD. For each region the specific mesh size can be freely chosen. Thus, by adding each contact face as a mesh region and assigning smaller specific mesh element sizes, the mesh will be locally refined at the contacts.

For executing this, the script iterates through the global list of all validated contacts, created by the function that finds all contacts (see section 4.3.4). As the list also contains the faces involved in the contacts, they can be directly added to a mesh region, additionally specifying the name of the mesh region and the characteristic mesh size of that region. The characteristic mesh size is set in VirSat and exported into the contact validation files, where the script can access it. It is important to note that, unlike the mesh groups, the mesh regions must be added before the actual mesh is generated due to the reason that they directly influence the meshing algorithm. Thus, if a mesh is created manually, the script detects it and the regions will not be generated for this mesh. However, when manually crafting the mesh, the engineer shall refine the areas he thinks make sense to refine, consequently the automated refining is unnecessary and also not possible to use in that case.

4.3.7 Processing Further Input Data

4.3.7.1 Emissivity

As soon as the mesh is generated and all mesh groups are created, the pre-processing can enter the next phase where the material parameters and boundary conditions are assigned to the mesh groups. First, the *setFaceEmissivities*() function is called. This function reads the *.rd files created by VirSat for each component and assigns the emissivity parameters for each element face.

First, the file is opened and every line is processed in that way, that the face number and emissivity values, which are separated by a comma, are added to a list. Face 0 hereby represents all faces of the component. If a face is specified apart from face 0, then the general emissivity value is overridden for that particular face.

Then, an iteration through all faces of the component starts, using the *getccxVolumes*() command to obtain the mesh element faces that are located within the current component face. If the component face does not exist in the list of component faces, the default value of face 0 is assigned to all the mesh faces by writing all according mesh face numbers and the emissivity values for each of it into the created radiation file. This is repeated for all faces of the component.

Face absorptivity, also being part of the *.rd file, is processed in the context of assigning solar and Earth albedo radiation to the model. This is described in section 4.3.8.

4.3.7.2 Heat Flow Boundary

After emissivity is assigned, heat flow boundary conditions are processed and assigned next. There are two types of heat flow boundary conditions, volume and face heat flows. For volumetric heat fluxes, the exported *.bfl file from VirSat is opened. There is only one number in it, the total heat flux acting on the component. This number is then divided by the volume of the component's mesh to represent the volumetric heat flux and then written together with the mesh group of the component as volumetric heat flux assignment back to the *.bfl file. To make sure that this operation is not done twice, when executing the script twice for example, the length of the file is checked. If there is more than one value, that means it was already processed and the process is aborted.

For face heat fluxes things are slightly more complex, since these fluxes cannot be assigned to the whole component but single element faces must be identified. One *.ehf file is generated for each component by the according VirSat app. For each defined face heat flux on that component, the file contains a line with that face number and the heat flux, separated by a comma. The script iterates through all lines, obtains the corresponding mesh element faces according to the method described in 4.3.7.1, and assigns the heat flux to all element faces located in the component face.

4.3.7.3 Temperature Boundary

The last boundary conditions to be set are the temperature boundaries. As for the heat flux boundaries, face and body temperature boundary conditions are accepted. However, the body temperatures were already assigned by the export app from VirSat. Thus, only the face conditions has to be processed. The assignment is similar to the face heat flow. One *.bcf file for each component is generated by VirSat, containing all information about the temperature boundary conditions to be set. There are now two possibilities for each line. Either there is only one value in the line, or there are two values separated by a comma. If there is only one value, this means a volume boundary was assigned for the whole component. Accordingly, the value is assigned to the complete component mesh group. If there are two values, then a face temperature was assigned. In that case the first value specifies the face and the second value specifies the temperature. Analogously to heat flux boundaries, the mesh element faces are collected for a component face and the temperature value is assigned to them. For this, the *Boundary* keyword together with either the according node numbers or the mesh group, the specification what type of boundary it is (”11” represents temperature), and the temperature value itself.

4.3.8 External Factors

With completing the assignment of the temperature boundary conditions, all system-internal boundary conditions are assigned. But, as described in chapter 3.3.3, there are external factors to be considered. These factors are Sun, Earth albedo and Earth infrared radiation.

4.3.8.1 Sun Radiation

To take the solar radiation, usually the most important aspect of the thermal environment of a spacecraft, into consideration for dynamic analyses, one has to assign a varying thermal load on the whole model. The magnitude and cycling of solar radiation depends on the orbit as well as the attitude of a spacecraft. To obtain this data, specialized software such as STK, General Mission Analysis Tool (GMAT), or similar can be used. For the following work, it is assumed that three required CSV files are supplied. One file contains the Sun vector with respect to a fixed coordinate system at every time step of the simulation. The second file contains the intensity percentage of solar radiation at every time step, such that it is zero in eclipse and 100 when Sun is in view. The third file contains the vector to the center of the Earth and the reflection under which a sun ray is reflected by Earth surface, or atmosphere, to the spacecraft. In appendix B.6 it is explained how the structure of these files shall look like.

As known from the sections before, loads were usually assigned to whole component faces, such that it was relatively convenient to obtain the mesh element faces within that component face and assign the same load to all of these faces. This is not possible in this case. If a component has a curved face, the load on the single mesh faces within this component face differs depending on the actual orientation of the mesh element face. Consequently, the load has to be calculated for every single mesh element face individually. Besides this, the orientation of the mesh faces cannot be obtained as a simple command, it has to be determined using different methods, described in the following.

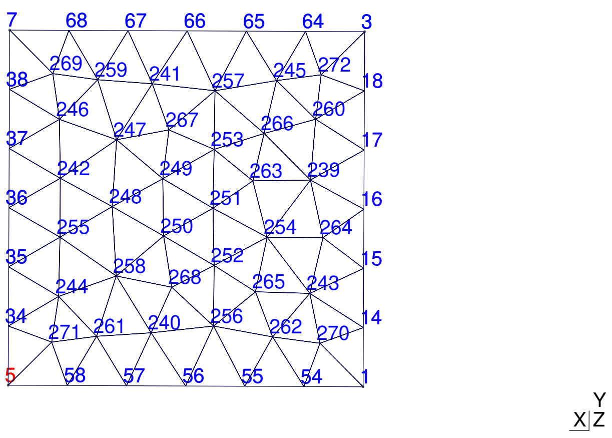
Now, to calculate the power input by radiation on a surface, one needs the Sun vector $\vec{s}$ and the face normal vector $\vec{n}$. From these two vectors the cosine of the angle β between them is calculated as the Sun intensity factor *f* that determines the Sun intensity on each face due to its own orientation, as seen in 4.1. The actual thermal load in $\mathrm{W\,m^{-2}}$ to apply on the surface then calculates as in 4.2 with α being the absorption coefficient, *f* being the mentioned factor for misalignment of the face, and *S* being the solar constant [31]. The equation is enhanced by with the solar visibility factor *i*, which represents the availability of sunlight at a certain time.

$$f = cos(\beta) = \frac{\vec{s} \cdot \vec{n}}{|\vec{s}| \cdot |\vec{n}|} \tag{4.1}$$

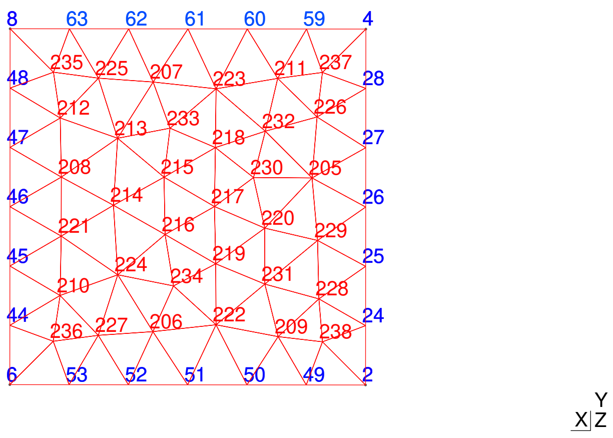
$$P_S = \alpha \cdot f \cdot i \cdot S \tag{4.2}$$

Sun vector, solar constant, and surface absorptivity are known but the face normal must be determined before the calculation can be executed.

To determine the face normal vector algebra is used. At least three nodes define each face. By subtracting the coordinates of node 1 from those of node 2 and node 3, respectively, one can obtain two vectors, representing two edges of the face. Calculating the vector product of the two vectors, a vector normal to the face is the result. Still, it is not certain if the vector points to the outside of the component or to the inside. This can be determined exploiting the fact, that in Gmsh the nodes of a face are always numbered counter-clockwise when looking at it from outside of the body. Thus, when projecting the nodes of two opposing faces of a body to the x-y-plane, the nodes of the one face are always numbered clockwise, while those from the other face are always counter-clockwise. Thus, in the x-y-plane as it is depicted in Fig. 4.19, the negative pointing (green) faces are always numbered counter-clockwise. When taking the vector product of vector



(a) Mesh of the component face pointing in positive (z-)direction, as seen from negative (z-)direction. Source: Own representation.



(b) Mesh of the component face pointing in negative (z-)direction, as seen from negative (z-)direction. Source: Own representation.

Figure 4.19: Meshes of two opposite oriented faces of a cube with node numbers displayed in Gmsh. The mesh input file entries where the nodes are numbered are provided in Tab. 4.2.

Table 4.2: Excerpt from the mesh file of the meshed cube. The upper part in the table shows three face elements of the face oriented in positive direction, the lower part three face elements of the face oriented in negative direction. When considering Fig. 4.19, the opposing direction in node ordering is visible.

Element Number	Node Numbers in order
562	247, 249, 248
563	250, 258, 248
564	251, 252, 250
482	215, 216, 217
483	215, 217, 218
484	217, 219, 220

1, that always points counter-clockwise, with vector 2, that always points clockwise, gives a normal vector that will always point outside of the body according to the right hand rule. The same holds for the face in positive direction, where vector 1 always points in clockwise and vector 2 always points counter-clockwise.

This is implemented in the *writeAmplitudeFile*() function. The function is executed once for each component and reads the previously created mesh file. It creates an amplitude, that is a discretized course of a varying load with the load being specified at each time step. For each time step, the value in the assigned amplitude is then multiplied by the base load value.

With the *getccxVolumes*() command all mesh elements and their corresponding mesh faces on a specific face of the component are obtained. For each mesh element face, the nodes forming this face are obtained from the mesh file. According to their sequence of occurrence among the 10 nodes in the line where the element is defined, it can be determined which node is the first, which the second and which third node of the face. The sequence depends on the face number of the mesh face within the mesh element and is specific for each type of mesh. For example, in a tetrahedral mesh, face number 2 is formed by node numbers 1, 4 and 2, in that order. This is the reason, why this workflow can only handle tetrahedral meshes. After the node numbers are determined, the next step is to obtain the coordinates of these precise nodes. The three nodes are then passed to the *findMeshFaceNormal*() function that calculates the two vectors and their vector product to return the normal vector. Then, the cosine of the angle between the face normal and the Sun vector is calculated for each entry in the Sun vector CSV file. This factor is then multiplied by the Sun intensity, absorptivity and the solar constant of 1367 $\mathrm{W\,m^{-1}}$ to calculate the resulting load on the mesh faces. Then, the additional radiation sources are added to this.

4.3.8.2 Earth Albedo

Albedo radiation, that is reflected by Earth, eventually hitting the spacecraft also introduces a heat flow load on the spacecraft faces. This load is smaller than the magnitude of the solar radiation itself, but still not negligible. It is not assumed as constant, as the constellation of the three bodies has large impact on its magnitude. However, the actual fraction of the sunlight that is reflected by Earth is assumed as constant, although it depends on local Earth surface and atmosphere composition [24].

After calculating the solar load, the FreeCAD script calculates this albedo load for each timestep. Analogously to the direct solar radiation, for each timestep the angles between the face normal and Earth vector are determined. As the albedo strongly depends on the constellation between Sun, Earth and spacecraft, this has to be taken into account as well.

Consequently, the course of the Earth vector over time has to be supplied in the same way as the Sun vector is. In addition, the (cosine of the) angle of reflection between spacecraft, Earth surface and Sun has to be supplied, as this influences the albedo intensity. In this work, the approximate resulting albedo load in $\mathrm{W\,m^{-2}}$ is calculated as described in 4.3, with α being the absorptivity of the face in the visible regime, β being the angle between face normal vector and Earth vector, γ being the reflection angle of the Sun rays reflected on Earth surface, R_{Earth} and R_{SC} being the radius of Earth and the height of the spacecraft over the center of Earth, a being the albedo factor of Earth (assumed as constant 0.35 [31]), S being the Solar Constant [31]. It is extended by i, the visibility factor of the Sun at that timestep.

$$P_{EA} = \alpha \cdot cos(\beta) \cdot cos(\gamma) \cdot \left(\frac{R_{Earth}}{R_{SC}}\right)^2 \cdot a \cdot S \cdot i \tag{4.3}$$

4.3.8.3 Earth Infrared

Earth's own infrared radiation is assumed as constant 300 $\mathrm{W\,m^{-1}}$ although there are some local variations in it, depending on the actual surface temperature and atmospheric composition [17, 24]. It is calculated in the same way as the solar heat flux, with the only difference that Earth is always present and the distance from spacecraft to Earth must be considered. Thus, no Earth visibility factor is needed and the resulting equation for the approximate heat flow in $\mathrm{W\,m^{-2}}$ on the spacecraft is described in 4.4, where α_{IR} now is the absorptivity in the infrared regime, β is the angle between face normal and Earth vector, P_{Earth} is the Earth-emitted radiation power per square meter, R_{Earth} and R_{SC} are the radius of Earth and the height of the spacecraft over the center of Earth, respectively [31].

$$P_{IR} = \alpha_{IR} \cdot cos(\beta) \cdot P_{Earth} \cdot \left(\frac{R_{Earth}}{R_{SC}}\right)^2 \tag{4.4}$$

If the cosine of the angle between face normal and Earth vector is greater than zero, the resulting infrared radiation load is calculated as stated in 4.4.

After the three radiation sources are calculated for each mesh face and each timestep, the resulting load for that timestep can be calculated by adding the three values up. This overall value is then written to a line the ”Amp.inp” file, with the elapsed time in seconds. After all timesteps were calculated for a mesh face, a load with the value 1 $\mathrm{W\,m^{-1}}$ is assigned with the before defined amplitude. To complete the file, these steps are repeated for every single mesh face.

4.3.9 Exporting Data from FreeCAD

After the temperature boundary conditions are assigned, all data from VirSat is processed. Furthermore, the files for the components’ meshes and boundary conditions are in the correct format to be read by CalculiX. Together with the files created in VirSat and the amplitude files representing the external loads, the simulation setup is now complete.

4.4 Feedback of Simulation Results to Virtual Satellite

After successful execution of CalculiX an output file is generated. This file contains setup data as well as result data. A shortened version is provided in listing 4.4. To close the loop of the workflow, this data must be transferred to the system model. Therefore, two actions are to be performed. First, the output file must be interpreted in that way, that the nodes and their corresponding temperatures are extracted from the output file. Second, the obtained data must be fed into the system model. To perform both actions, the VirSat app *ReadSimulationOutput* was written. The flow chart of the app is provided in Fig. 4.20. First, the system model is accessed to obtain the information if it is a steady-state or a transient analysis. This is important, as for transient analyses the process of obtaining all nodes and temperatures has to be repeated as many times as there are timesteps. The number of timesteps is one if it is a steady-state analysis, or the specified number of timesteps for a dynamic analysis.

Listing 4.4: Condensed version of an output file of a (static) simulation. The dots indicate where parts are left out. After general simulation information all nodes are listed, then all elements with their respective nodes are displayed. Finally, all nodes with their temperatures and all elements with their heat flows in each direction are listed.

```
[Simulation Information]
...

 -1          1-1.00000E+00-1.00000E+01 4.99000E+02
 -1          2-1.00000E+00-1.00000E+01-1.00000E+00
 -1          3-1.00000E+00 0.00000E+00 4.99000E+02
 -1          4-1.00000E+00 0.00000E+00-1.00000E+00
...
...

 -1          1    6    0    1
 -2         86       2375          7        601       3760       3759        102        806       3761        807
 -1          2    6    0    1
 -2       2375        170          7        601       3762        187       3759       3761        808        807
 -1          3    6    0    1
 -2       2099        487        546        586       3764       1527       3763       3765       1529       1528
 -1          4    6    0    1
 -2        443        592       2321        570       1018       3766       3767       1017       1019       3768
 -1          5    6    0    1
 -2        494       3667        674        557       3770       3769       1601       1599       3771       1600
...
...

 -1          1 2.97828E+02
 -1          2 2.98004E+02
 -1          3 2.97757E+02
 -1          4 2.97966E+02
...
...

 -1          1 8.59588E+02 3.67043E+01 3.86994E+01
 -1          2-3.03529E+02 1.39403E+02-1.72789E+02
 -1          3-1.31599E+03 2.61523E+03-1.42618E+02
 -1          4-5.92680E+02 8.81888E+02-2.63265E+02
...
...
```

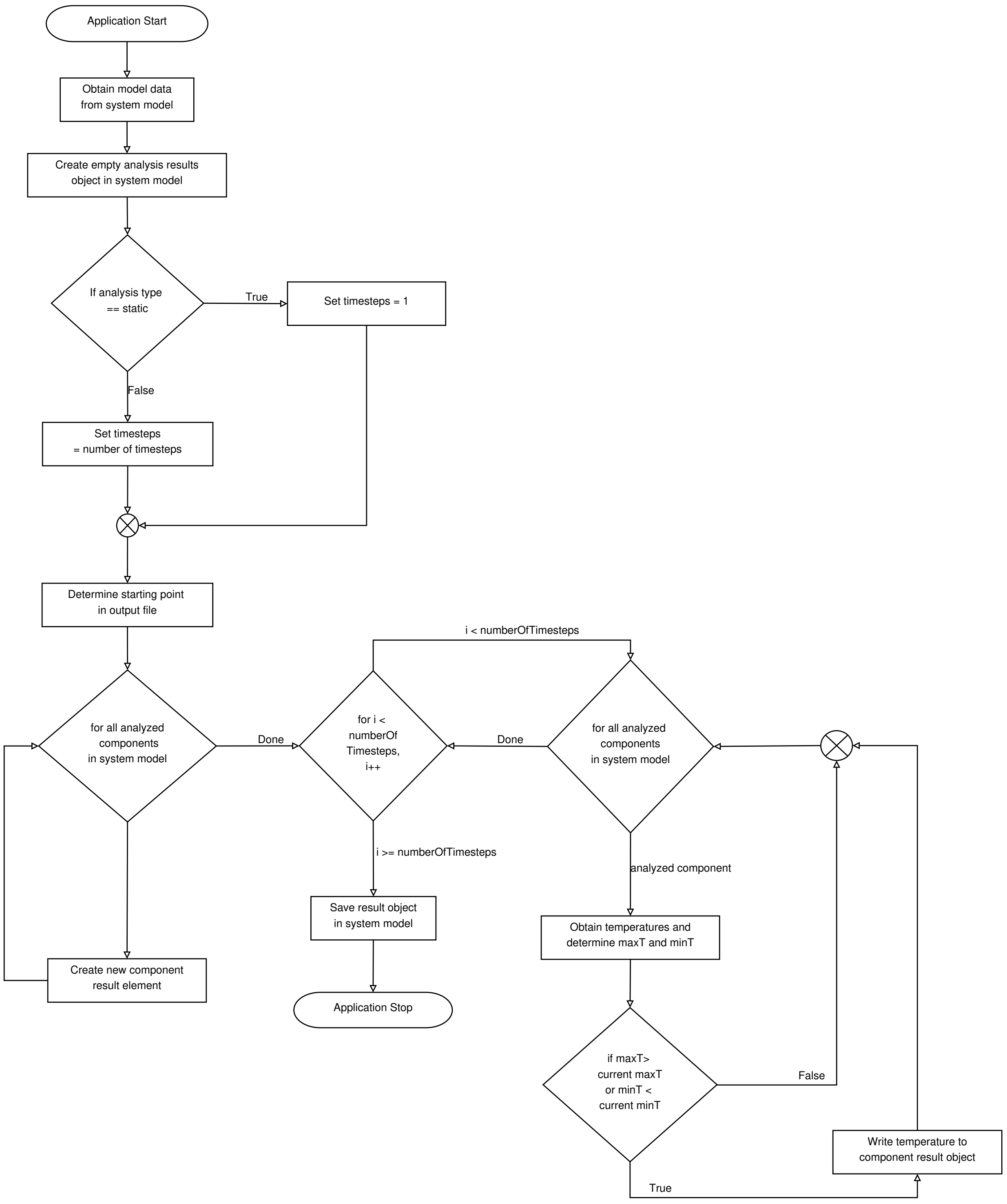

Application Start
Obtain model data from system model
Create empty analysis results object in system model
If analysis type == static
True
Set timesteps = 1
False
Set timesteps = number of timesteps
Determine starting point in output file
for all analyzed components in system model
Create new component result element
Done
for i < numberOf Timesteps, i++
i < numberOfTimesteps
i >= numberOfTimesteps
Save result object in system model
Application Stop
Done
for all analyzed components in system model
analyzed component
Obtain temperatures and determine maxT and minT
if maxT> current maxT or minT < current minT
False
True
Write temperature to component result object

Figure 4.20: Flowchart of the "ReadSimulationOutput" app that reads the output file and creates the "AnalysisResult" element in the system model. Source: Own representation.

Then a for loop iterates through the first lines of the output file to identify the line where the first node is defined. The line number depends on the number of components and contacts and therefore has to be determined every time the app is executed. This line then represents the app-internal starting line of the file.

Afterwards, an iteration through all components is executed to obtain some data necessary to execute the rest of the program. In the iteration loop, for each component the corresponding mesh file is opened, where the number of elements and nodes was deposited during execution of the FreeCAD script. These two numbers are obtained and then added to the total number of nodes and elements, respectively. Furthermore, the number of nodes of each component is saved in an array for later use. Finally, each iteration is completed by creating an element to store the results in the system model for each component, so that the obtained temperatures can be saved there.

Then the main part of the program starts, for each timestep, a sequence of actions is performed resulting in the accumulation of all node temperatures at all timesteps. For this, in each timestep an iteration through all components is executed. Using the fact, that the sequence of components as it is iterated in VirSat is the same sequence the components and their nodes are represented in the output file (in the end, the component sequence of the whole workflow is determined by this iteration sequence, as the same iteration is used in the export apps), only the number of nodes for each component is necessary to identify those lines, where the nodes and temperatures for each component are located. Starting from the before determined starting line, the first line where a node and a temperature is named is two times the total number of elements, plus the number of nodes and a fixed value of seven, that represents the blank lines. As mentioned, an iteration through all components is executed. Within this iteration, all temperatures belonging to the node numbers are interpreted and for each component the maximum and minimum temperature is evaluated and stored in an array, overwriting the old maximum or minimum temperature in case they are higher or lower, respectively. If it is the first timestep, the maximum and minimum temperature are always overwritten, to replace the default value, which has no physical meaning. After iterating through all components, all node temperatures of that timestep have been collected and the outermost iteration loop through the timesteps starts again, provided that there is more than one timestep.

As soon as the iteration of the outer loop is completed, the elements containing the maximum and minimum elements are instantiated in the system model and the app is executed successfully.

Chapter 5

Thermal Analysis Evaluation

5.1 Checking Validity of Results

To ensure the workflow can be used by engineers in real projects, it has to be validated that the used solver delivers valid results and that the workflow itself produces a valid simulation setup that matches the system model. For validation purposes the established thermal analysis software ANSYS is used in parallel to CalculiX. The obtained results of CalculiX with its standard solver Spooles and ANSYS are compared to judge the accuracy of the solution. The examined model is designed to represent an object similar to a satellite and is depicted in Fig. 5.1. Different shapes are employed to test the automatic geometry processing of the work flow as complete as possible.

5.1.1 Solver Validation

The solver is validated using the same input setup for both thermal analysis tools. If the results of CalculiX match those of ANSYS, the solver can be accepted as validated. To validate the results of the solver as good as possible, all features that can be used in the workflow shall be present in the examined models. Only that way it can be ensured best that the solvers obtain the same

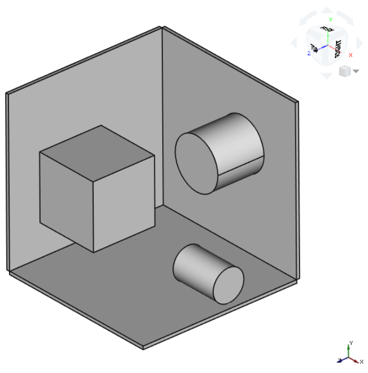


Figure 5.1: Geometric representation of the model to be analyzed. Source: Own representation.

Table 5.1: Aspects of the thermal model to be considered in validation activities

Heat transfer
Conduction
Contact conduction
Radiation
Boundary conditions
Temperature (face)
Temperature (body)
Heat flux (face)
Heat flux (body)
Simulation Properties
Steady-state
Dynamic
Model properties
Different materials
Different face radiation properties

results, regardless of the model. Tab. 5.1 lists all features provided by the workflow, which are to be considered. To avoid having to include all types of boundary conditions in one single model, and due to the fact that with static and dynamic simulation at least two simulation setups have to be executed anyways, two boundary conditions will be applied to the static model and the other two to the dynamic one.

5.1.1.1 Static Case

First, the static case is examined. The boundary conditions are distributed to the two simulations as mentioned, with volumetric heat flow and face temperature being applied to the static one. The model is shown in Fig. 5.2, where the face highlighted in green is the face where the fixed temperature of 300 K is applied to (”Science Payload”) and the volume in green is the component where the volumetric heat flow of 30 W in total is assigned to (”Battery”). A detailed setup of the thermal model is provided in appendix C.1.

The first model consists of 21744 nodes and 10947 elements. The model for CalculiX is fully created with the automated workflow, while for ANSYS, parts of the model have to be created manually. Mesh, mesh groups, contacts, materials, and material assignments can be directly imported from the CalculiX model, but all boundary conditions (including radiation properties) have to be set manually.

In Fig. 5.3 the results of the two solvers are visualized in their software-specific post-processor. The maximum and minimum values (obtained from the output files) differ by 0.13 K and 0.00 K, respectively. Taking the result of ANSYS as reference that corresponds to 0.038 %, and 0 % differ-

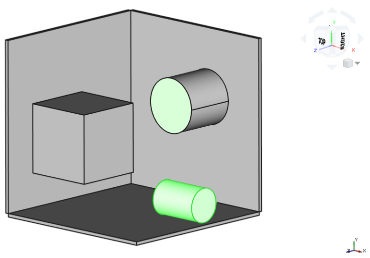
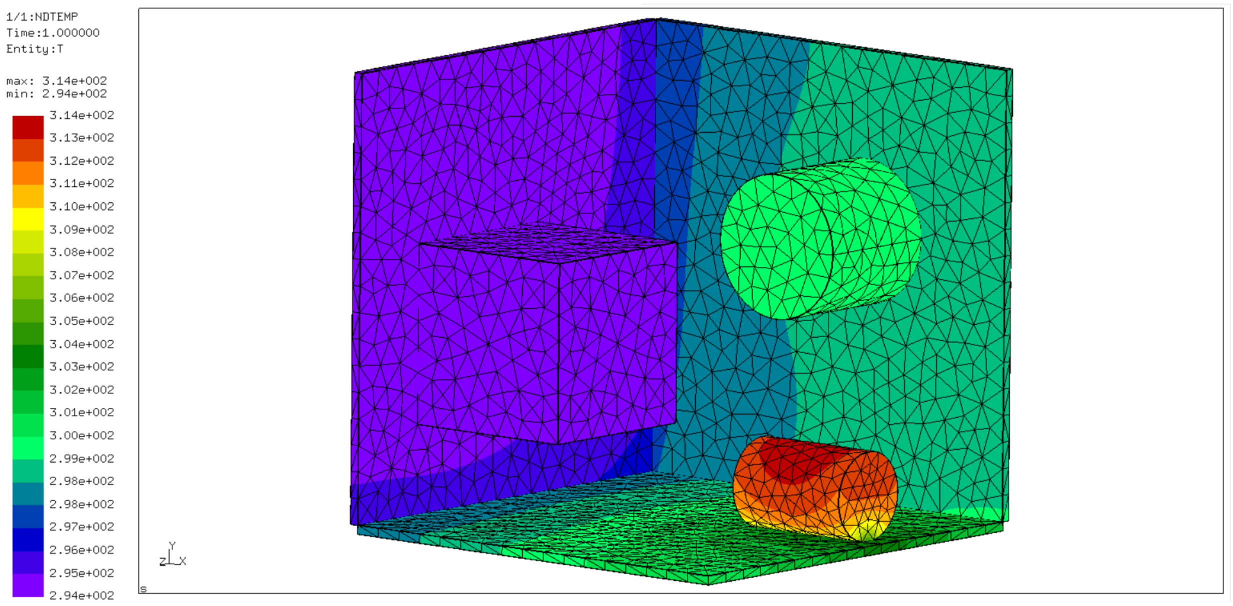
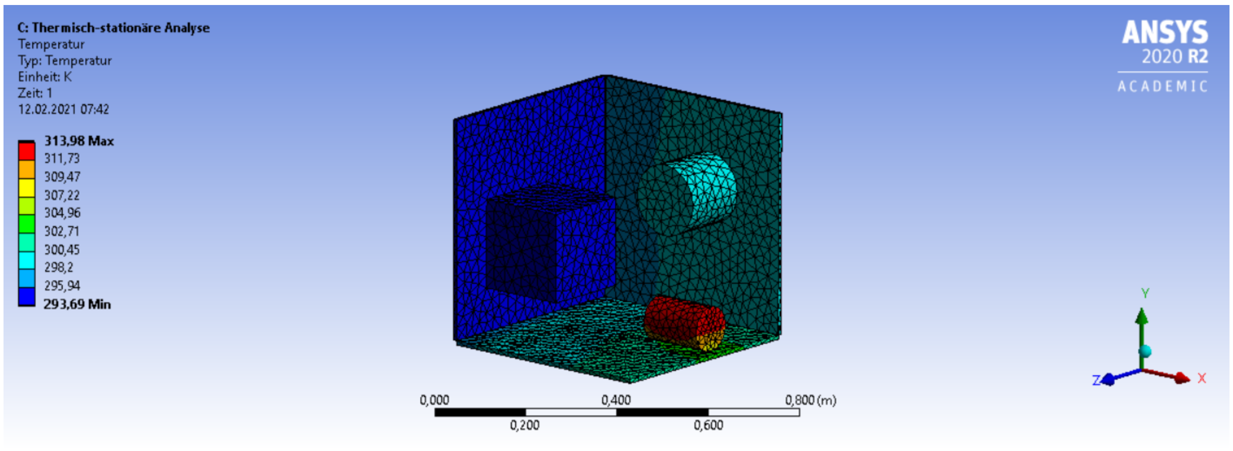


Figure 5.2: The model to be analyzed. The green highlighted face is where the temperature boundary is set, the green body where the volumetric heat flow is applied. Source: Own representation.



(a) Result of the first simulation as seen in the CalculiX GraphiX post-processor. Source: Own representation.



(b) Result of the first simulation as seen in the ANSYS post-processor. Source: Own representation.

Figure 5.3: Graphical results of the two simulation tools, as shown in their respective postprocessors.

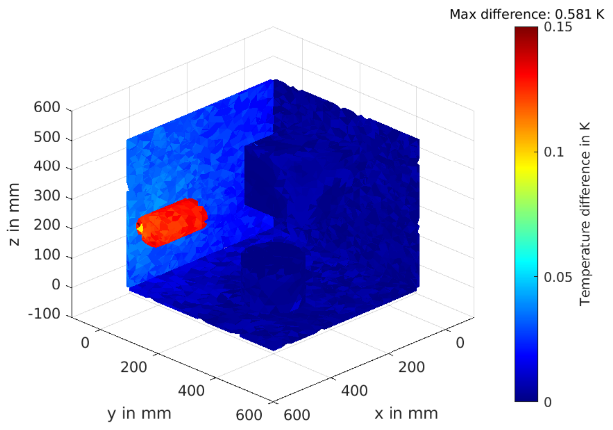
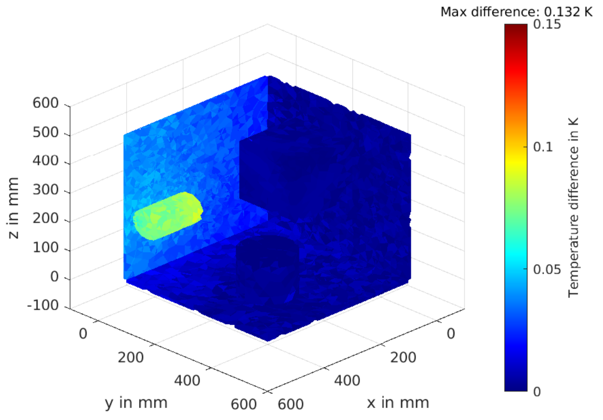


Figure 5.4: The deviations between the solvers mapped to the mesh. The maximum deviation of 0.581 K occurs on the contact area between the lying cylinder and the baseplate. Source: Own representation.

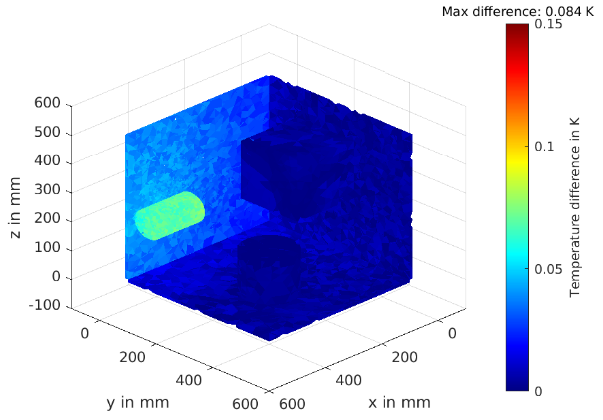
ence. The results were further examined using a Matlab Script that reads in the nodes and elements of the simulation model as well as the two simulation output files. The scripts subtracts the CalculiX result from the ANSYS result and feeds the absolute of this value to an array. This array is then plotted by assigning each value to the respective node and display the results in a three-dimensional mesh. Fig. 5.4 shows this plotted mesh for the first model. It is visible that the maximum difference between the simulation is actually in the order of 0.581 K. This means there are nodes with a higher error than the node with the highest temperature. However, in the array of error values itself one can see that the extreme value is indeed 0.581 K, but there are only a total of 69 nodes in the whole model that have a deviation larger than 0.15 K. The rest of the model, as follows from Fig. 5.4, shows very small errors compared to the extreme values, which indicates that there is some problem exclusively at the location of the high error.

To further examine this problem the location of this error was tracked and it was found that only nodes on the contact between the heated cylinder and the base plate show this high difference. Although the difference is still low compared to the absolute temperatures, the model was checked thoroughly and some parameters were adjusted to find possible errors that cause this behavior. Finally, the mesh was refined locally on the point of contact between these two components. This increased the number of nodes from 21744 to 24491 and lead to a decrease of the maximum deviation to around 0.132 K. The corresponding mapped errors are depicted in Fig. 5.5a and the changed model setup is listed in appendix C.2. What is conspicuous, is that the error from the point of maximum temperature decreased as well (to around 0.12 K), although the amount of nodes and elements was only increased in one specific location. This shows that a fine mesh in critical areas can improve the results significantly and can influence the whole model, not only on that specific locations.

After achieving such an increase in result quality, the mesh at the contact and the mesh of the cylinder were further refined to test if the error would further decrease at the same rate. For this,



(a) Deviation between ANSYS and CalculiX solutions of the second static model, mapped to the individual nodes. Source: Own representation.



(b) Deviation between ANSYS and CalculiX solutions of the third and last static model, mapped to the individual nodes. Source: Own representation.

Figure 5.5: Differences between ANSYS and CalculiX results plotted to the mesh.

the contact mesh was refined even more to 5 mm maximum specific length, while the cylinder mesh as such was set to 15 mm maximum specific length. The node number increased to 34815 and the simulation results showed a further decrease in error to a maximum deviation of 0.084 K between ANSYS and CalculiX models. This is depicted in Fig. 5.5b. The corresponding model details are listed in appendix C.3.

5.1.1.2 Dynamic Case

Validation of the dynamic simulation is done with the remaining boundary conditions. Consequently, a face heat flow of an arbitrary value, here 1380 $\mathrm{W\,m^{-2}}$, is applied to the highlighted face in Fig. 5.6 and a volume temperature boundary with 300 K to the highlighted component (Battery). Apart from this, the model is identical to the one in the static case. Again, the complete simulation setup is listed in appendix C.4. For this part the mesh with 24491 nodes, that was used in the second simulation of the static case is utilized. The total time was set to 100 s with each timestep covering 1 s, resulting in a total number of 100 timesteps. Another two error maps are created with the Matlab script, as it was done in the static case. One for an intermediate timestep (50 s) and one the last timestep (100 s). The resulting errors are shown in Fig. 5.7a and 5.7b. It is visible that small errors again arise, and again they are located in the same contact areas as before. The errors are still quite small after 100 s but have the tendency to increase. For a long term simulation this might become significant, however a finer mesh would decrease the error most likely, as it was concluded in the static case.

Regarding the results of static and dynamic simulations, the solver can be considered as validated. The results are very close to each other, especially when the mesh size is sufficiently small. With all these simulations still having rather coarse meshes compared to actual thermal analysis meshes,

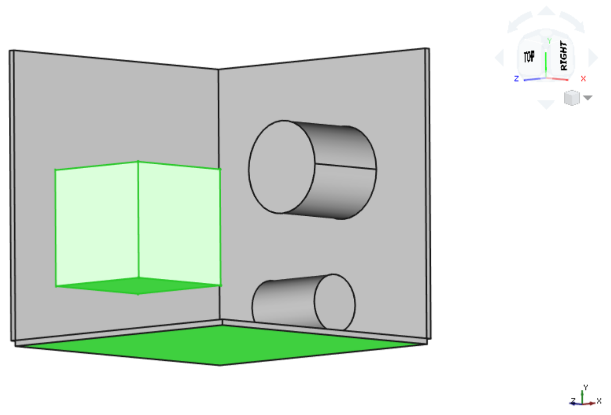
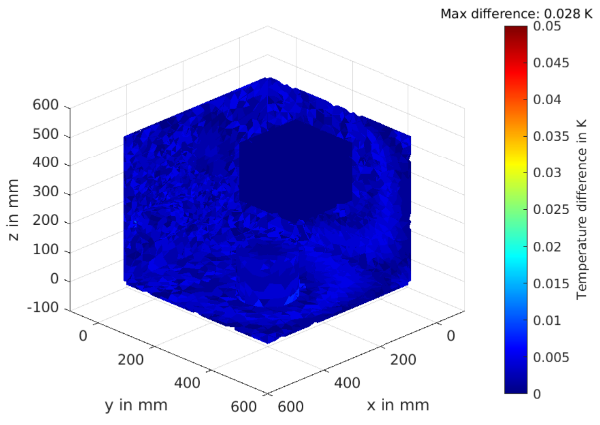
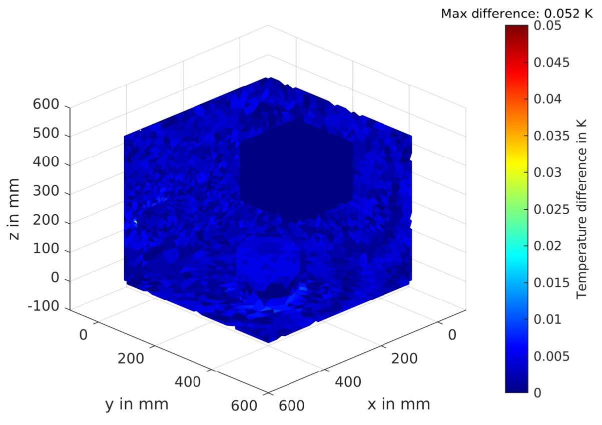


Figure 5.6: The model to be analyzed. The green highlighted face is where the heat flux applied to, the green body where the body temperature condition assigned to. Source: Own representation.



(a) Deviation after 50 s (50 timesteps). Source: Own representation.



(b) Deviation after 100 s (100 timesteps). Source: Own representation.

Figure 5.7: Differences between ANSYS and CalculiX results, mapped to the mesh. The maximum deviations always occur near contacts.

the deviation between the results can be expected to become even smaller if the mesh node and element number is further increased.

5.1.2 Workflow Validation

To validate the general workflow created in this thesis as such there are essentially two possibilities. Either a model of a spacecraft is created for which a thermal analysis already exists, to compare the obtained results to the actual results of the thermal analysis. Or two identical models are created independent of each other. One in VirSat that experiences all aspects of the workflow, and one crafted manually in ANSYS, assigning all boundary conditions utilizing the ANSYS GUI. This way it is validated that the boundaries set in FreeCAD by the created script are correct. If the results match, the workflow is proven to generate the thermal simulation model correctly. The model for this should be as ”challenging” as possible, especially regarding the processing of the geometry. For this, the different forms and the rather uncommon contact from the model used before in the solver validation are used.

The meshes used for this validation are identical for both models, as different meshes would influence the results and therefore it would not only be a validation of the workflow itself. Validation of the mesh is a different topic, that is not covered here as it does not necessarily concern the workflow as such.

To consider all aspects that are part of the workflow in the validation, again two models, a static and a dynamic, are to be examined. For reasons of simplicity, the same models as in section 5.1.1 are used for this validation and the mesh with 24491 nodes is chosen. A benefit of using the same model is that the results can also be compared to the ones obtained before. As the whole input for CalculiX used before was already generated using the automated workflow, it can be reused, so that only the model for ANSYS needs to be created. For this, only the bare mesh files of the components, not containing anything except the nodes and elements, and the main input file, only containing the invocation of the mesh files is imported in ANSYS. Thus, everything except the mesh has to be created in ANSYS.

5.1.2.1 Static Case

In the first iteration of this validation step, significant differences between the simulations were detected. After examination, the reason for this was found in the way the contacts are defined. In ANSYS there are two possibilities to define the contact faces. ANSYS creates solid bodies out of the imported mesh. The first option, which was initially used, was to select the faces of these bodies that are in contact with each other as the contact partners. This lead to some deviations between the results in CalculiX and ANSYS (> 2 K). As this was unreasonably high compared to the results before, an investigation was performed. The conclusion of this is, that the contacts are calculated in a different way, when choosing the actual meshes’ faces instead of the restored solid bodies’ faces for the contact definition. With the actual mesh faces selected as contact partners,

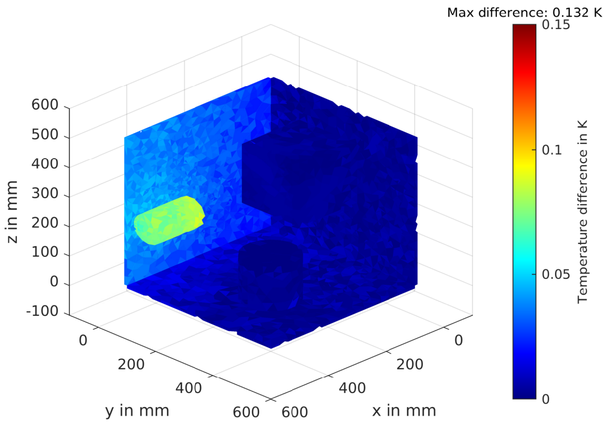


Figure 5.8: The deviations between ANSYS and CaculiX model occurring in the static workflow validation. The values are identical to the ones from Fig. 5.5a. Source: Own representation.

the results are exactly matching the before obtained result in ANSYS from section 5.1.1. Consequently, they are matching the results obtained in CalculiX to around 0.132 K. The fact that this way the results match the expectations indicates that there is an error introduced when ANSYS restores the geometry from the mesh.

5.1.2.2 Dynamic Case

With the contact areas defined correctly, also the manually created dynamic simulation in ANSYS shows exactly the same results as the partly automated ANSYS simulation, that was executed in section 5.1.1. Here, only the result after 60 timesteps is considered. The maximum deviation between the automated CalculiX model and the manually created ANSYS model is 0.052 K, as depicted in Fig. 5.9. This is again the same value as in the solver validation before. The highest deviation occurs on the contact locations, which indicates once more that the refinement of the mesh in these critical areas is of high importance.

With these results obtained, it is safe to say that the workflow applies all boundary conditions and materials correctly. This was proven with a hand crafted model analysis setup in ANSYS, that showed the same results as the partly automatically generated ANSYS setup in the solver validation, and very similar values to the fully automatically generated setup in CalculiX.

5.1.2.3 Validation of Sun Intensity

In this workflow, an option to generate Sun intensity data over the course of a specified time is provided. As ANSYS does not provide a similar option, the correctness of the calculations cannot be validated in the same way the other parts of the workflow are. In fact, only few, commercial tools offer the option to include orbital influences to the thermal analysis. Thus, a different approach is chosen for validation. All external orbital data is assumed to be valid, since it is gener-

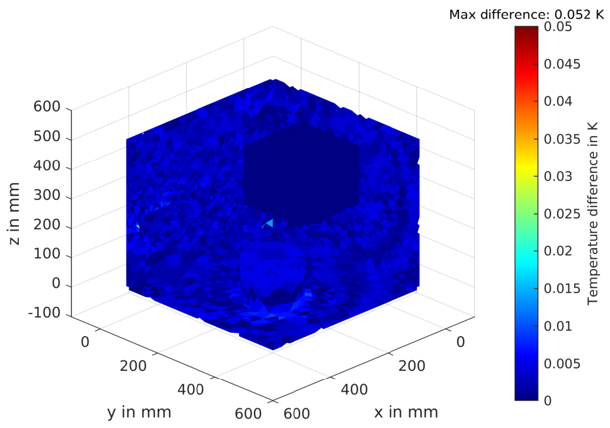


Figure 5.9: The deviations between ANSYS and CaculiX model occurring in the dynamic workflow validation after 100 timesteps (100 s). The maximum value is identical to the one from Fig. 5.7. Source: Own representation.

ated using the commercial tool STK and not as part of the workflow. The correct assignment of the resulting loads is checked by visualizing the load on the faces for one timestep, comparing the direction of the Sun vector with the calculated loads in the model. Fig. 5.10 shows the mesh of the model with the Sun vector at the initial timestep. From this, the calculation of the Sun intensity is performed, resulting in an intensity factor between 0 and 1, that is displayed for each element in Fig. 5.11. Due to the fact that it is only possible to assign values to elements of a three-dimensional mesh, not to the individual faces, faces close to the edge often show wrong values, as they show the value that was last assigned to the element. Often this is the wrong value from another face in the element that points in a different direction. However, away from the edges it can be clearly seen which value is actually prevalent in that area. To ensure this, such a face is selected as a sample and the actual calculated value of Sun intensity for that face and its neighboring face is obtained from the "Amp.inp" file.

The face that is chosen for this is located on the far upper right corner of the cube in Fig. 5.11. It is displayed as light blue there, indicating an intensity value of around 0.3. The face, just as the one around the corner, belongs to element number 9980 and its face number within this element is 2. With this information one can now search the "Amp.inp" file, to find out the actual values assigned to the face. For the timestep in the picture, the value 0.885 is assigned, proving that the face actually has a right value assigned, and the value displayed in Matlab just shows the value from the other face of the element (which is 0.249).

Apart from the faces close to the edge, all faces pointing in the same direction show the same colors and those oriented towards the sun show the highest values. This indicates that the assignment of the sun intensity works as intended, still it is certainly no proof for this.

With the effort spent here, it was possible to validate the sanity of the values that are obtained by this function. However, to fully validate the results, it is necessary to compare them to reference

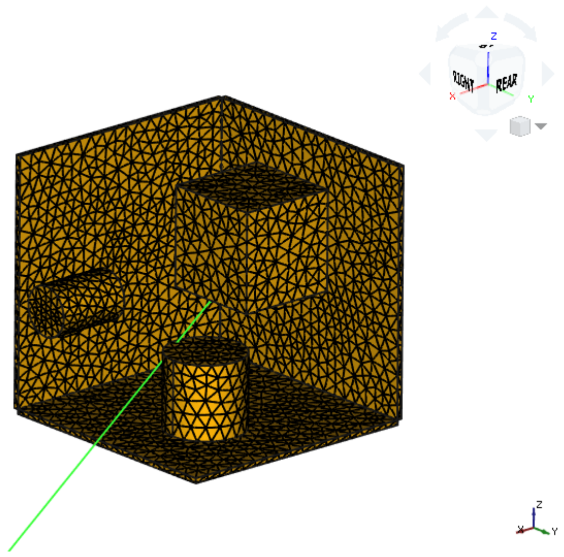


Figure 5.10: Mesh of the model with a representation of the Sun vector at the same timestep as Fig. 5.11. Source: Own representation.

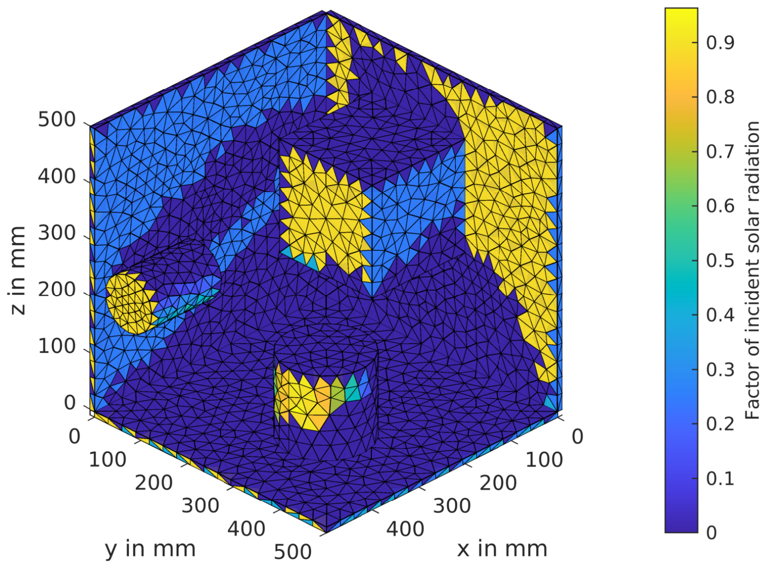


Figure 5.11: Distribution of the Sun intensity factors over the mesh faces. Note: Mesh faces close to edges are often displayed in a misleading way, since only one value can be applied to an element that has two faces on a surface. Source: Own representation.

values. Due to the lack of software offering such values, and the lack of time to generate such values in a different way, this kind of validation is the only one that can be provided within this work.

5.1.2.4 Validation of Earth Infrared and Earth Albedo Calculation

To round off the orbital radiation influences, albedo and infrared radiation from Earth were included in the FreeCAD script. Although the magnitude is lower than radiation from Sun, it is still significant. Calculation of Earth infrared loads is basically done in the same way as the Sun radiation loads, with the difference that the Earth vector is used instead of Sun vector and that there is no visibility factor as Earth is always visible. Also the magnitude is smaller, thus instead of the Solar Constant a value of 250 $\mathrm{W\,m^{-2}}$ is used. Apart from that the calculation is exactly the same. Due to the large degree of equality in calculation of the actual value and the complete equality in calculating the involved vectors and angles, the application of Earth infrared radiation is assumed as equally valid as Sun radiation.

Earth albedo is applied in the same way, only that the calculation is slightly different. As the nature of the albedo being the solar radiation reflected by Earth, the constellation between Earth, Sun and spacecraft is important for the intensity of the albedo. Thus it includes an additional term for this relation, and the visibility modifier that tells if Sun is visible or not. For validating the applied Earth albedo load, the same holds as for Earth infrared radiation. Due to the high degree of similarity in calculating and applying the load, as well as the underlying equation coming from a trusted source, the validity of Earth albedo application is assumed as equally valid as Sun radiation application.

5.2 Improvement of Result Quality

As seen before, the results obtained are already rather close to each other and the solver and workflow are validated to a certain degree. However, there is always room for improvement and that also includes this workflow. It is clear, that automatically generated meshes will not deliver results as accurate as the ones obtained with handcrafted meshes by experienced thermal engineers. Still, by decreasing the element size of the mesh, either locally or globally, the result will come rather close to simulations set up manually with great effort.

In some cases it might also be more beneficial to examine the model using a hexahedral mesh or an entirely different mesh type. This is not covered in this work, however could be implemented in future ambitions with relatively little effort.

Not only the mesh influences the result quality. Other factors as the amount of detail, correctness of the boundary conditions, material parameters, radiative parameters geometries also determine the validity of the result.

5.3 Limitations of the Provided Workflow

The workflow presented in this work is an efficient way of building a thermal model within a system model, analyzing it and feeding the results to the system model. The sections before showed that there is some small deviation when comparing the results from CacluliX to those of ANSYS, however when comparing the results obtained with the workflow with those obtained completely manually, there is no difference. Still, there are cases where the usability of this workflow in the provided form might come to its limitations. This could be the case for example, when the system model in VirSat is highly detailed and a lot of very small components are in the model. For the level of detail of the thermal analysis this is beneficial, but if the orbital radiation influence feature is used, the computing time will increase drastically due to the resource-intensive boolean operation when detecting the obstruction of a face. This boolean function is a bottleneck that increases processing time of the FreeCAD script from a few seconds to minutes or hours, depending on the size of the model. The thermal properties of such small components might not be relevant for the whole system and could therefore potentially be neglected in the thermal model. To ignore these components of the system model in the thermal model, it is only necessary to not add a "ThermalData" element to them. The workflow will then simply ignore this component. The decision if a component is important for the thermal model or not must be evaluated by the thermal engineers.

Another limitation for the workflow is the mesh. It is only possible to use a generic tetrahedral mesh. This is a general purpose mesh that can fit complex geometries well, however for simple geometries that are used in VirSat hexahedral meshes might be favored over the tetrahedral meshes. If so, the FreeCAD script would have to be extended slightly to enable the use of other meshes, at least if the user wants to include the orbital radiation in a dynamic simulation. This was also described in section 4.3.8.1.

The workflow itself does not introduce a limitation for part complexity. However the complexity of the parts is only determined by the geometries that can be imported in VirSat. With future releases this might improve.

As mentioned, the main weakness of this workflow is the long computing time for the orbital radiation influence. With this only being due to one single command, that takes some time for execution, there might be a way for a workaround to make the execution of the script faster. In addition, the calculation of the orbital influence is only done with some simplification. For all three types of radiation the values are considered as constant all over Earth's surface. However, in reality there are variations [17]. Finally, what is also ignored is the radiation incident on a face due to reflections on other components. To model all these factors appropriately, substantially more effort would have to be spent than it was possible in this thesis. It also has to be taken into account, that due to high extra effort, as well as additional required software packages, it was not possible to really validate the orbital radiation feature. The values obtained seemed to be sane, but could not be compared to reference values.

Chapter 6

Outlook

Although this work provides a workflow and tools to execute a complete thermal analysis, there are still many aspects to be considered before referring to the adaption of thermal analyses in MBSE as complete. This work lays ground for this, but cannot cover all aspects to consider, as the topic is simply too complex to be fully completed in such a short period of time. Nevertheless, the necessary base functions and some extras are already implemented, the workflow does work and the suggestions made in this section are more of complementary nature rather than required additions.

With the results obtained with the automated workflow matching those obtained with a manual simulation in ANSYS to a high degree, it is safe to say that reasonable results can be achieved with this workflow. Still, a full validation requires a lot more effort than it was possible to spend in this work. This especially holds for the inclusion of the orbital radiation influences. In this work, their validation was done by judging the results qualitatively. However, this is not enough for ensuring correct results. Thus, a full validation using a FEA software that also supports the orbital radiation influences would be beneficial.

The aspect that has most potential for improvement is the mesh. In this work only generic meshes were used to establish the workflow. As the type of mesh as well as the mesh itself strongly depend on the model to examine, more flexibility for using different types of meshes would be useful in this workflow. As mentioned before, technically different mesh types are supported by the workflow, however to enable the use of different mesh types together with the orbital radiation feature, small additions in the FreeCAD script would have to be made. Another aspect is the analysis of the mesh size itself. To check convergence of the mesh, multiple simulations with varying mesh sizes have to be compared to each other. In the current state, for each simulation the mesh sizes of all components in the system model have to be changed accordingly, then the workflow has to be executed again. Here, an automation of this process would be beneficial to quickly analyze the convergence of the mesh.

When this workflow is sufficiently mature, a tool like RCE, that allows to create simulation workflows including multiple software packages, could be used to further automate it. This way, the user would not have to open the different tools manually, but would just have to start the workflow while the remaining actions would be sequenced and executed by RCE.

While for an experienced user, most of the potential errors occurring might be relatively easy to track down, an app that checks the consistency and integrity of the thermal model within VirSat would be another useful addition, as it would help especially new users to find errors easier. Some aspects this could potentially cover would be detecting missing parameters or spotting thermal ports that are not assigned to a thermal interface.

Another aspect that has potential major improvement is the geometry modeling. VirSat itself offers only the possibility to create the basic geometrical shapes. Of course, not every shape can be modeled sufficiently with this. Here, boolean operations would significantly enhance the geometrical modeling capabilities. While there is, in theory, the option to import geometries in *.stl format, this feature was often producing bugs, and was therefore not used in this work. However, in future versions this feature might be improved, introducing much better modeling capabilities as well. It is pointed out here, that the workflow itself in theory does not impose any restrictions on the complexity of geometries. Still, if used with complex geometries, some testing for sanity would be necessary as this could not be performed in this work.

For possible further improvement of the result quality, or smoothening of the workflow by decreasing computation times, the different solvers CalculiX offers could be evaluated for the best suited one. Spooles is described as a fast, but has no multi-core capability. In addition it is limited by the random-access memory (RAM) capacity [42]. Thus, another solver might be better suited for larger models, while still delivering valid results.

Regarding post-processing, the present app that only determines maximum and minimum temperatures for the components, could be improved by introducing further functions. For example, in dynamic simulation, differences between the temperatures across timesteps could be calculated to determine the rate of temperature changes during the simulated time.

Furthermore, all parts of the workflow could be adapted in the concurrent engineering version of VirSat, as it requires relatively little information, that can also be approximated, about the system to work, and is executed quite fast. Thus, it works with low effort. This could help the thermal engineers in concurrent engineering studies to evaluate different thermal control concepts. As all apps currently address specifically the configuration tree, this would have to be changed to project tree, if the workflow is intended to be used in the earliest phases. It would just be an adjustment in the source code that is done quite fast.

Finally, the original intention was to model Compasso as a part of this work to conduct validation with this model. However in accordance with the project team it was decided that this task would be scrubbed. One reason for this is that there was not sufficient valid data readily available to model Compasso appropriately and the second reason was that the whole validation process would have been very exhaustive with a higher number of components. Instead, an example model for a payload was used to validate the solver and the workflow. In hindsight, this was the better option, as a more complex model would have made the error tracing more difficult in the validation process. Still, modeling Compasso would be a good first application of the provided workflow and is therefore encouraged to do as soon as the project is advanced enough for this.

Chapter 7

Conclusion

Fully adapting MBSE in all aspects of spacecraft development is a complex undertaking. Many different models exist, many different analyses with interdependencies have to be executed. Still high initial resource need in this case leads to significant time and resource saving potential. In this thesis, a workflow is proposed to execute thermal analyses of a space system, using data from the system model, stored in the MBSE tool VirSat. For this, the ”thermal” concept was established in VirSat, providing a blueprint for the storage of data that is necessary for the execution of thermal analyses. This concept covers a range of mandatory data for thermal analyses as well as some optional additions like mesh size parameters.

In addition to the thermal concept, a number of apps for VirSat were written to support thermal engineers in extracting relevant information from the system model. The apps are part of the workflow and either convert system model data directly to the input format of CalculiX, or to an intermediate format that can be read by a FreeCAD script, which is also part of this thesis, to eventually form a complete input set for the CalculiX FEA software.

The mentioned Python script for FreeCAD is utilized to automatically mesh the geometry exported from VirSat and apply all boundary conditions and material properties to the correct subsets of the mesh. The combination of the VirSat apps and the FreeCAD script essentially cover the whole preprocessing process and the result of their execution is a complete simulation setup. As an extra feature specifically for space systems, the FreeCAD script also provides the option to include an externally obtained Sun vector, and Sun visibility data to model a time-dependent load on each mesh face, that together represent the total solar radiation loads on the spacecraft. In addition, similar data can be used to model the in-orbit heat flows on the spacecraft caused by Earth albedo and Earth infrared radiation.

Finally, this thesis offers an app that allows VirSat to interpret and add the simulation result data to the system model. This is enabled by obtaining maximum and minimum temperatures throughout the simulation output for each component individually.

The simulation results obtained with the workflow were examined by comparing them to the results obtained with a manually, independently created thermal model with the commercial FEA software ANSYS. The deviations were quite small, indicating that the workflow creates the correct thermal model. Lacking a suitable tool to model the orbit radiation influences, it was not possible to include this feature in the validation efforts. However, a qualitative evaluation was executed and the results appear reasonable.

Appendix A

CalculiX Input Script Description

A.1 Geometric Input

In a CalculiX input file the geometry must be provided as a finite element mesh. This means the geometric model is not represented by different simple parametric shapes, but it is meshed beforehand and therefore represented by the nodes and elements of the mesh that represent the geometry. Before each node, element, or mesh group is defined, the name of the specific set has to be set. This is important as material assignment, contact surfaces and volume loads, to name some, are introduced using these set names.

Every node is defined by its node number and three cartesian coordinates in one line, separated by commas. Elements, which are composed of nodes, share a similar syntax. Analogously to node definition, the element number is defined first, then the numbers of the nodes that form the borders of the element are enumerated. Again, these numbers are separated by commas.

The third concept used in the geometry shall be called mesh group. A mesh group is a named collection of nodes or elements that represents specific features of the geometry, for example a single surface. Mesh groups play an important role in assigning all boundary conditions, surface parameters and contact areas correctly. One variation of a mesh group is the surface mesh group. This set is basically a regular mesh group with element numbers, but each line also references a specific face of the called element. This way a surface can be represented by the equivalent mesh element faces.

If the same set name is used more than once, the existing set is not replaced but extended by the nodes or elements that are in the second set.

A.2 Thermal Input

After specifying the geometry, the thermal information is written to the input file. It includes the initial temperature, contact conductivities, material properties, surface properties and, finally, boundary conditions.

The initial temperature is set with the keyword *Initial Conditions*. For a static analysis the initial temperature has no influence on the results but must be specified anyways. In dynamic simulations it has much more influence. For setting the initial temperatures, the nodes have to be called using the appropriate node set name, then the temperature is set, separated by a comma from the node set name. It is important that all nodes have an initial temperature allocated. If one node has more than one temperature assigned, the last assignment overrides all previous ones.

After setting all initial temperatures the material properties are set. In CalculiX there is no direct assignment of material parameters to elements. Instead, first a certain material is defined with the *Material* keyword. Then, the name, thermal conductivity, specific heat capacity and density are set for the material. The defined material with its parameters is afterwards allocated to the corresponding elements. It is most desirable to create one element set for each component to assign a material to the complete component by just assigning it to the respective element set once. With the *Solid Section* keyword followed by the material name and the element set name, a material is assigned to an element set after it is defined. This is repeated until all elements have a material allocated.

Following the material properties, the areas of contact have to be described for CalculiX to interpret and apply them correctly. A contact description in the sense of a CalculiX input file generally consists of two parts: The definition of the contact properties and the involved mesh faces. To define the contact properties *Surface Behavior* is called after assigning a label to the contact with *Surface Interaction* keyword. There are multiple possible parameters to set for the contact behavior. However, for thermal analysis most of them are not used as they mainly describe mechanical behavior. Thus, they do not have to be set. Still, one mandatory information, even if not relevant for the thermal analysis, is the pressure-overclosure relation stating the type of this relation and its slope. Linear pressure overclosure means the relation between pressure and overclosure between the contacts follows a linear growing function while an exponential behavior is also possible. In any case, this does not affect a purely thermal simulation as there are no mechanics involved whatsoever. Thus, the value set for the slope does not influence the result of the simulation. What does influence the result of the simulation, in contrast, is the *Gap Conductance* keyword, which specifies the thermal conductivity across the contact and is set after the surface behavior. Furthermore, a temperature and pressure can be set. This only has an influence, if the value of contact conductance is desired to be set variable for varying temperature and pressure values and is therefore of no further interest here.

To complete contact specification, the user now has to assign two contact faces to the already defined contacts. This is done with the *Contact Pair* keyword, succeeded by the contact label and the contact type, which is in this work, for reasons of interchangeability of the contacting surfaces and compatibility, always *Surface to Surface* a contact. With the other choice being *Node to Surface* type it is not allowed to use both types within the same simulation setup. Thus, for the sake of easier processing, all contact faces simply will be provided as surface element sets instead of having to differentiate between nodes and surface elements. To complete the contact specification, in the line below the before mentioned settings, the two surface element sets are listed, separated by a comma. The contact is now completely defined for CalculiX being able to calculate with it.

When all contacts are specified, the boundary conditions are the next information to be written to the input file. In a CalculiX thermal analysis, the user can set six types of them:

1. Convection
2. volume distributed heat flux
3. face distributed heat flux
4. concentrated heat flux
5. radiation
6. temperature.

Convection being negligible in space, thus for spacecraft only of interest if used as a closed system, is ignored in this work, just as concentrated heat flux on single nodes, which would be not practical as the user would have to select single nodes for this. If necessary, it is still possible to add a concentrated heat flux manually.

Radiation assignment uses the *Radiate* keyword. To set a specific surface as a radiator, as well as absorber, the radiation parameter is assigned to each individual mesh element face. In addition to the single elements themselves, it has to be specified if the residual radiation when view factor is lower than one is radiated towards an environment node (*CR* - cavity radiation) [42]. If CR is set, the environment temperature must be specified for CalculiX being able to calculate the amount of radiation coming from the environment. If this temperature is set negative, the view factors are scaled to one instead of radiating to (and from) the environment node. As in space most of the radiation is radiated to the environment node with the temperature of the cosmic background radiation, this is a fixed setting for all simulations conducted. The syntax of the *Radiate* command is that the actual face number of each element is wrapped with an *R* and the optional *CR* to one expression (e.g. *R*2*CR*). This expression follows the element number and is followed by the temperature of the environment node and the magnitude of the emissivity.

Volume distributed heat flux and face distributed heat flux are set by using the *Dflux* keyword. It is followed by stating the element list or element numbers and, if applicable, the appropriate face of the element for a face distributed heat flux. Finally, giving the value of the heat flux density either as heat flow per volume for volume distributed heat flux or as heat flow per area for face distributed heat flux, respectively. The distributed heat fluxes can represent incoming solar radiation for example.

The last usable boundary condition is the temperature boundary condition. It is set with the *Boundary* keyword with the actual specification following in the next lines. First, the node number or set of nodes the boundary condition shall be applied to is referenced. Separated by a comma, the types of boundary condition are specified. In this case, boundary condition 11, which is the temperature boundary condition is chosen. Lastly, and again separated by a comma, the temperature value is handed to the input file.

A.3 Simulation Configuration

For setting up a simulation one has to specify some general things about how the simulation shall be executed. Accordingly, there are some commands that have to be employed for CalculiX to properly set up the simulation. As some of the simulation specifications have to be set at a certain point of the input file, some of the commands are actually written in between other commands of the previous subsections. As a first general information, some physical constants for thermal calculations have to be specified. The physical constants actually have to be fixed before the *Step* keyword, because after this only boundary conditions and post-processing are specfied. The constants are set by calling the keyword *Physical Constants*. To conduct a thermal analysis two physical constants are essential for the solver. First, the temperature of absolute zero in the desired unit to enable correct calculation of the resulting radiation emission and heat flows in all units that have a constant offset from Kelvin. Second, the Stefan-Boltzmann constant which is one parameter of 2.8, used to calculate radiant fluxes.

Moreover, as CalculiX is also a software for mechanical simulation, it has to be stipulated what kind of simulation it shall execute. This is done right after the *Step* keyword. To conduct a thermal simulation, the keyword *Heat Transfer* is used. In the very same command the type of thermal simulation, steady state or transient, is assigned. It is separated by a comma from the *Heat Transfer* keyword.

The last information in the input file addresses post processing. Here the user can specify which output is desired. CalculiX can create a file with all nodes (*Node File*) or all elements (*El File*) and their respective parameters. Parameters of most interest for thermal calculations are the node temperatures (*NT*) and the node or element heat fluxes (*HFL*). The desired parameters are listed below the node or element file keywords.

Appendix B

Step By Step Manual to Conduct a Thermal Analysis with the Provided Workflow

B.1 Step 1 - Modeling the system

1.1 First, a new project is created in VirSat. In the repository, the concepts *Core*, *ProductStructure*, *Visualization* and *thermal* must be added from the registry. In addition, the apps feature must be activated by clicking on *Activateapps* in the repository.

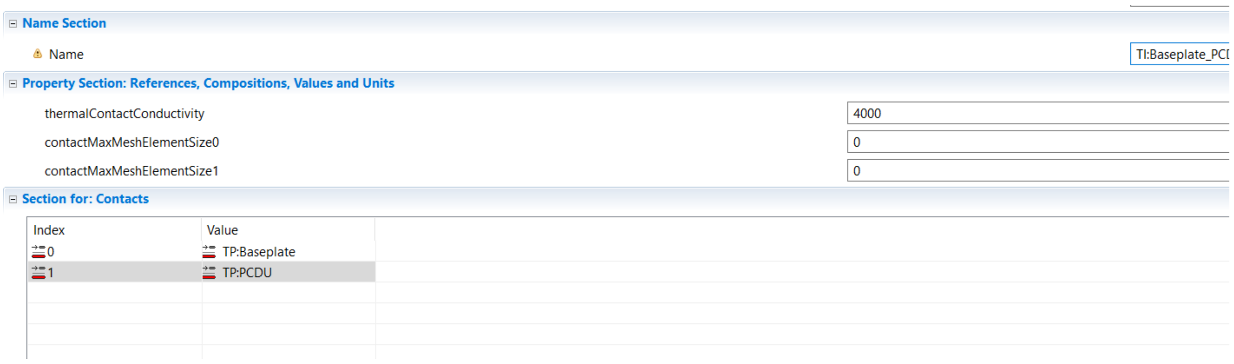
1.2 To import the necessary apps, they either can be added by drag and drop in the project explorer from another project, or the content of each app is copied to a newly created app, which is then renamed accordingly (Project Explorer -> apps -> right click on app -> *refactor*-> *rename*)

1.3 Then, the model is created. Start this by creating a configuration tree, add the desired subsystems and their components. Add the visualization for each component and arrange it accordingly. Note that overlaps should be prevented when not intentionally used for generating a contact area as described in the "How To" in appendix B.6.

1.4 If not already present, create the *ThermalControlSubsystem* (or similar name). Right click on the subsystem to add the *ThermalAnalysis* element. Then add the *MaterialCollection* to this element. Now there are two options: - Import materials from an external .csv file. For this, place a Materials.csv file in the project folder (Workspaces/runtime-virsat_thermal/DemoProjectManual), containing the desired materials in the format (*NameWithoutSpacesAndUnderscores*, *ThermalConductivity*$[W/m/K]$,*HeatCapacity*$[J/kg/K]$,*Density*$[kg/m^3]$,*emissivity*,*absorptivity*). Then execute the *ImportMaterials* app. This will add the specified materials to the material collection - Manually create materials. Right-click on *MaterialCollection* to add a *Material* element. Then specify all parameters of this material. Note that emissivity refers to IR-emissivity, while absorptivity refers to absorptivity in the visible range. The value chosen for powerBalance will introduce a distributed volume boundary condition on the component.

1.5 For each component with a visualization, that is to be considered in the analysis, add a *Thermal-Data* element by right clicking on the component element configuration and selecting *Add Thermal-Data* under *thermal*. Inside the *ThermalData* element create a *ThermalElementParameters* element. Type in the required parameters and link the material to one of the imported/created materials.

1.6 In the *ThermalAnalysis* element, create a *ThermalContacts* element in which a *ThermalPortList* and a *ThermalInterface* are created. In the *ThermalPortList* specify one *ThermalPort* for each component (and name it TP:XXX), linking the component by referencing to the *ThermalElementParameters* element of that component. In the *ThermalInterfaceList* create one *ThermalInterface* for each contact between two components. Reference the two components' thermal ports (in the *Contacts* section) and specify the thermal contact conductivity of the contact. If desired, specify also the maximum mesh sizes for the contact areas. An example on how this could look like is provided in Fig. B.1. NOTE: If the two contacting components are overlapping each other, this will be detected at a later step. The component specified first will cut the overlap out of the component specified second.


Name Section

Name	TI:Baseplate_PCI

Property Section: References, Compositions, Values and Units

thermalContactConductivity	4000
contactMaxMeshElementSize0	0
contactMaxMeshElementSize1	0

Section for: Contacts

Index	Value
0	TP:Baseplate
1	TP:PCDU

Figure B.1: View on how a thermal interface is specified. Source: Own representation.

1.7 If the contacts and visualizations are specified, execute the *WriteContacts* app. And export the geometry using the *Cad Export Wizard* under *Files->Export* to a desired location.

1.8 Start FreeCAD and choose the *VirtualSatellite* workbench. Import the before exported geometry using the appropriate button as depicted in Fig. B.2.

1.9 Make sure, the path used in the *prepareModel* script is the one where the contact files from VirSat are exported to. Then execute *prepareModel* to obtain the set of new faces for each contact. NOTE that ANY face number specified in VirSat later on refers to these face numbers, NOT the default ones (This script is basically identical to the *makeContactFaces*() function of the *Script-_ProcessModelToInput* script and therefore the face numbers used in the main script are generated in the *prepareModel* script.)

1.10 In the *ThermalAnalysis* element, create a *BoundaryConditions* element. For each temperature

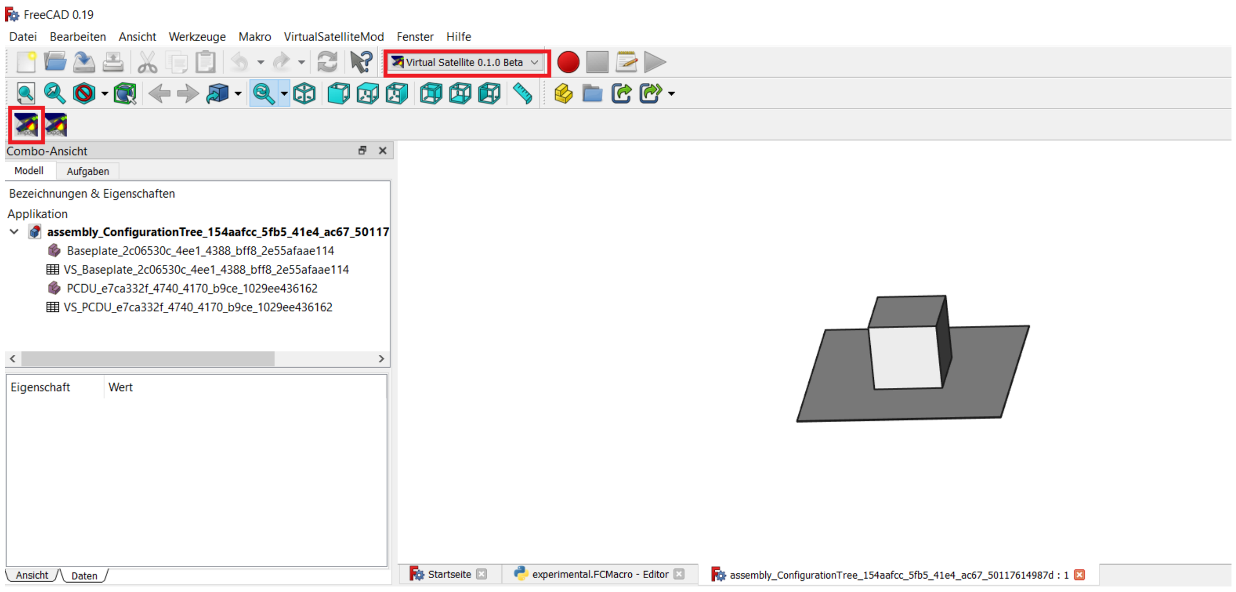

FreeCAD 0.19
Datei Bearbeiten Ansicht Werkzeuge Makro VirtualSatelliteMod Fenster Hilfe
Virtual Satellite 0.1.0 Beta
Combo-Ansicht
Modell Aufgaben
Bezeichnungen & Eigenschaften
Applikation
assembly_ConfigurationTree_154aafcc_5fb5_41e4_ac67_50117
Baseplate_2c06530c_4ee1_4388_bff8_2e55afaae114
VS_Baseplate_2c06530c_4ee1_4388_bff8_2e55afaae114
PCDU_e7ca332f_4740_4170_b9ce_1029ee436162
VS_PCDU_e7ca332f_4740_4170_b9ce_1029ee436162
Eigenschaft Wert
Ansicht Daten
Startseite
experimental.FCMacro - Editor
assembly_ConfigurationTree_154aafcc_5fb5_41e4_ac67_50117614987d : 1

Figure B.2: View on the FreeCAD window. Import button and workbench dropdown menu are highlighted in red. Source: Own representation.

boundary and face heat flux, create a corresponding element inside. Then reference the relevant component to apply the boundary condition, and the value of the boundary condition (in K or $\mathrm{W\,m^{-2}}$, respectively). If applicable, specify the face number of the desired face from the prepared model, as mentioned in the step before.

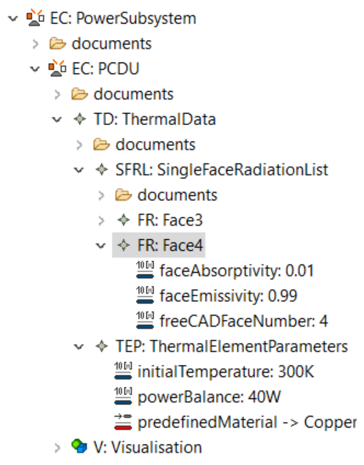
1.11 In the *ThermalAnalysis* element, create a *MeshSizes* element. If desired, for each component an individual maximum mesh element size can be specified by adding a *ComponentMeshSize* for each component, rename it after the component and then specify desired mesh size and reference the according components *ThermalElementParameters* element.

1.12 If desired, for each component the individual faces can have individual face radiation parameters to account for different surface properties. To do this, add a *SingleFaceRadiation* object to the components *ThermalData* element. Then add a *FaceRadiation* element for each surface that differs from the overall material value specified in the *MaterialCollection*. In this *FaceRadiation* element, absorptivity(vis), emissivity(IR) and the face number of the prepared model in FreeCAD are specified.

B.2 Step 2 - Preparing the Analysis

Now the system's thermal aspects are modeled completely. The following steps are executed to specify some simulation parameters and export the model from Virtual Satellite.

2.1 Inside the *ThermalAnalysis* element, create an *AnalysisType* element. Specify the type of simulation to be executed (Transient/Static). And, for transient simulations, specify the total time of



EC: PowerSubsystem
documents
EC: PCDU
documents
TD: ThermalData
documents
SFRL: SingleFaceRadiationList
documents
FR: Face3
FR: Face4
faceAbsorptivity: 0.01
faceEmissivity: 0.99
freeCADFaceNumber: 4
TEP: ThermalElementParameters
initialTemperature: 300K
powerBalance: 40W
predefinedMaterial -> Copper
V: Visualisation

Figure B.3: Example of a single face (Face4) that has individual radiation parameters assigned. Source: Own representation.

simulation and the time step size in between. Then choose if the inclusion of the orbit influences (such as solar radiation) is desired or not. If *Include* is chosen, the appropriate orbital data must be supplied. For more information on this, read the "How To" in appendix B.6. Note that choosing *Include* substantially increases processing time in FreeCAD, as the obstruction of each face has to be determined for each time step. The way this can be done in FreeCAD is very resource demanding and therefore the process can take severe amounts of time at this stage.

2.2 Execute all export apps having a *write* in their name, or execute the *combinedExport* app.

B.3 Step 3 - Processing with FreeCAD

The thermal model is now in an intermediate state, some aspects are ready for CalculiX but most have to be processed in FreeCAD. This part is almost completely automated, so if not desired the user does not have to do anything apart from executing the script.

3.1 Check if all exported files are present in the desired directory (There must be files with the endings .rd, .ehf, .bcf and .bfl for each component. In addition, the files MeshSizes.txt, Materials.inp, ValidateContactsMaster.txt, ValidateContactsSlave as well as the add_contact.inp and main.inp) and, if orbital radiation is included, the files "Sun_Vector.csv", "Solar_Intensity.csv" and "Earth_Vector.csv" must be provided, specifying the Sun vector, Earth vector (and cosine of reflection angle), and the solar intensity for each time step.

3.2 If not imported any more, import the CAD model again using the Virtual Satellite Workbench.

3.3 In FreeCAD, it is possible to create a custom mesh instead of using the automatically generated one. For this, create a mesh element and mesh the desired component. Then name the mesh element exactly after the component (name+_+ID). FreeCAD will then automatically add "001" behind this, which is fine. NOTE: By default, the script will reset the state of the FreeCAD document to delete all mesh elements and part elements it generates during runtime. To include a default mesh, deactivate the *reset*() line by adding an ”#” before it. And make sure that before executing the script, only the desired mesh elements and the default, imported parts are present! Otherwise the model will not be consistent.

3.4 Execute the FreeCAD script *Script_ProcessModelToInput*.

B.4 Part 4 - Executing Simulation

When the FreeCAD script was executed successfully, all input for CalculiX is present in the directory. Now the simulation can be executed.

4.1 Open *CalculiXlauncher*3.4 and choose *ccx*2.16 and *cgx*2.16 versions. Then open the main.inp file and click on *RunSolverCCX*.

B.5 Part 5 - Postprocessing

When the simulation ended successfully, the results are needed to be fed back to Virtual Satellite. For the user, it might be also convenient to view the simulation results graphically, using CGX by clicking on $RunCGXPost - ProcessorMode$. The simulation results were written to the main.frd file, which can be interpreted from the VirSat app *ReadSimulationResults*.

5.1 With the main.frd file in the directory, execute the ReadSimulationResults (make sure that all properties in the *AnalysisType* were not changed during the workflow, else the app will misinterpret the output file. It creates (if not already present) a *ThermalAnalysisResults* element in the *ThermalAnalysis* element. In it, a new *AnalysisResult* is created for each analysis. The app will obtain the maximum and minimum temperature value of the whole simulation, for every component.

B.6 How To?

Set all paths correctly?

It is important that all paths throughout all apps and scripts are set to the same desired path. Else, the files will be located wrong and cannot be found by the subsequent program that tries to access the file. The locations where paths have to be set are:

1. In all export apps (every path they contain)
2. In the prepareGeometry script in FreeCAD
3. In the processModelToInput script in FreeCAD (only the path right at the top)

Create a contact for two parts that have a contact without area?

Use the placement in the visualization to shift one component slightly into the other one. This way an overlap is created and if the contact is specifies in VirSat, FreeCAD will cut the overlapping piece out of the second component. This way a small contact face is created between the components.

Obtain the files necessary to use orbital solar (and Earth) radiation influence?

To obtain these files a tool like STK or GMAT can be used. In it, the orbit of the spacecraft can be modeled and the resulting Sun vector at the points of time relevant for the analysis can be exported. In addition, the obstruction of the Sun by Earth f.e. must be exported (in STK it is *Sun Intensity*. The files are to be supplied in the following structure:

For Sun_Vector.csv:
Header line
Timestep1, VectorX, VectorY, VectorZ (vectors relative to spacecraft fixed coordinate system)
Timestep2, VectorX, VectorY, VectorZ
...
An example for this is displayed in listing B.1

Listing B.1: Example of the structure of the Sun_Vector.csv file as it is generated by STK.

```
"Time (UTCG)","x (km)","y (km)","z (km)"
30 Dec 2020 11:00:00.000,130253576.662236,36557761.626092,−57757642.069708
30 Dec 2020 11:01:00.000,130411397.929910,35989165.559428,−57757510.174463
30 Dec 2020 11:02:00.000,130566743.587159,35419870.024707,−57757353.951172
30 Dec 2020 11:03:00.000,130719612.464795,34849886.353209,−57757173.580512
30 Dec 2020 11:04:00.000,130870003.412705,34279225.936515,−57756969.353358
30 Dec 2020 11:05:00.000,131017915.292355,33707900.224151,−57756741.669519
30 Dec 2020 11:06:00.000,131163346.969396,33135920.721108,−57756491.036008
30 Dec 2020 11:07:00.000,131306297.306429,32563298.985220,−57756218.064825
30 Dec 2020 11:08:00.000,131446765.155935,31990046.624426,−57755923.470279
```

For Sun_Intensity.csv:
Header line
Timestep1, Intensity in percent
Timestep2, Intensity in percent
...
For Earth_Vector.csv
Header line
Timestep1, VectorX, VectorY, VectorZ, cos(ReflectionAngle)
Timestep2, VectorX, VectorY, VectorZ, cos(ReflectionAngle)
...

FreeCAD does not adapt a changed geometry

The Virtual Satellite workbench in FreeCAD stores copies of the geometry files in the corresponding folder within the user files of FreeCAD. If a part is imported and a file with the same name is already present within this folder, the workbench will just take that file. Thus, all files with part names from the C:/Users/UserName/AppData/Roaming/FreeCAD/Mod/VirtualSatelliteCAD folder must be deleted in that case.

The visualization does not appear in VirSat

If there is trouble with the visualization concept and or the 3D viewer, save the model and restart VirSat. It should now show the visualization (provided the right project is selected in the 3D viewer).

In FreeCAD: ”ValueError: ... not in list”

No MeshSizes element was created in Virtual Satellite (1.11). Create the element, even if it is left empty, and export the model again.

In FreeCAD: ”The meshregion:... is not used to create the mesh because the CharacteristicLength is 0.0 mm.”

This is perfectly normal if you don’t have specified a mesh size for a contact. It only says that the mesh region was not created as there is no parameter for it.

Reduce the time FreeCAD needs to calculate the orbital radiation loads

Unfortunately, the boolean function used for determining if a face in FreeCAD is obstructed is quite slow. To make the execution of the script as fast as possible, it is recommended to cut the vector and intensity files at the last time step. Else, the script goes through the complete file and calculates the course of the radiation load for every timestep in the vector files (which should generally be either equal to or more than the specified timesteps for the simulation).

Appendix C

Simulation Setups

C.1 Base Model (Model Static 1)

The base model is the model used in the first validation simulation. Since all simulated models are quite similar, for all other models only the parameters differing from the base model are provided in their respective sections. Every other parameter will be as it is in the base model.

Simulation Type: Steady-state

Component 1: BaseplateXZ
Initial Temperature: 300 K
Characteristic mesh size: 30 mm
Material: Aluminum
Conductivity: 155 $\mathrm{W\,m^{-1}\,K}$
Specific Heat: 915 $\mathrm{J\,kg^{-1}}$
Density: 2700 $\mathrm{kg\,m^{-3}}$
Emissivity: 0.03
Heat Flow: None
Temperature Boundary Condition: None

Component 2: BaseplateXY
Initial Temperature: 300 K
Characteristic mesh size: 30 mm
Material: Aluminum
Conductivity: 155 $\mathrm{W\,m^{-1}\,K}$
Specific Heat: 915 $\mathrm{J\,kg^{-1}}$
Density: 2700 $\mathrm{kg\,m^{-3}}$
Emissivity: 0.03
Heat Flow: None
Temperature Boundary Condition: None

Component 3: BaseplateYZ
Initial Temperature: 300 K
Characteristic mesh size: 30 mm
Material: Aluminum
Conductivity: 155 $\mathrm{W\,m^{-1}\,K}$
Specific Heat: 915 $\mathrm{J\,kg^{-1}}$
Density: 2700 $\mathrm{kg\,m^{-3}}$
Emissivity: 0.03
Heat Flow: None
Temperature Boundary Condition: None

Component 4: PayloadScience
Initial Temperature: 300 K
Characteristic mesh size: 30 mm
Material: Aluminum
Conductivity: 155 $\mathrm{W\,m^{-1}\,K}$
Specific Heat: 915 $\mathrm{J\,kg^{-1}}$
Density: 2700 $\mathrm{kg\,m^{-3}}$
Emissivity: 0.03
Heat Flow: None
Temperature Boundary Condition: Face 3, 300 K

Component 5: PayloadCommunications
Initial Temperature: 300 K
Characteristic mesh size: 30 mm
Material: Copper
Conductivity: 400 $\mathrm{W\,m^{-1}\,K}$
Specific Heat: 385 $\mathrm{J\,kg^{-1}}$
Density: 8900 $\mathrm{kg\,m^{-3}}$
Emissivity: 0.03
Heat Flow: None
Temperature Boundary Condition: None

Component 6: PowerBattery
Initial Temperature: 300 K
Characteristic mesh size: 30 mm
Material: Titanium
Conductivity: 10 $\mathrm{W\,m^{-1}\,K}$
Specific Heat: 480 $\mathrm{J\,kg^{-1}}$
Density: 4800 $\mathrm{kg\,m^{-3}}$
Emissivity: 0.03
Heat Flow: Volume, 30 W, applied to the volume of the mesh in CalculiX and the volume of the

retrieved part in ANSYS (slightly differing)
Temperature Boundary Condition: None

Contact 1: BaseplateXZ to BaseplateXY
Contact Conductivity: 4000 $\mathrm{W\,m^{-2}\,K}$
Contact Partner 1 mesh size: 0
Contact Partner 2 mesh size: 0

Contact 2: BaseplateXZ to BaseplateYZ
Contact Conductivity: 4000 $\mathrm{W\,m^{-2}\,K}$
Contact Partner 1 mesh size: 0
Contact Partner 2 mesh size: 0

Contact 3: BaseplateXY to BaseplateYZ
Contact Conductivity: 4000 $\mathrm{W\,m^{-2}\,K}$
Contact Partner 1 mesh size: 0
Contact Partner 2 mesh size: 0

Contact 4: BaseplateXZ to PowerBattery
Contact Conductivity: 4000 $\mathrm{W\,m^{-2}\,K}$
Contact 1 mesh size: 0
Contact 2 mesh size: 0

Contact 5: BaseplateXY to PayloadScience
Contact Conductivity: 4000 $\mathrm{W\,m^{-2}\,K}$
Contact Partner 1 mesh size: 0
Contact Partner 2 mesh size: 0

Contact 6: BaseplateYZ to PayloadCommunications
Contact Conductivity: 4000 $\mathrm{W\,m^{-2}\,K}$
Contact Partner 1 mesh size: 0
Contact Partner 2 mesh size: 0

C.2 Model Static 2

Base model modified by:

Contact 4: BaseplateXZ to PowerBattery
Contact Conductivity: 4000 $\mathrm{W\,m^{-2}\,K}$
Contact Partner 1 mesh size: 10 mm
Contact Partner 2 mesh size: 10 mm

C.3 Model Static 3

Base model modified by:

Component 6: PowerBattery
Initial Temperature: 300 K
Characteristic mesh size: 15 mm
Material: Titanium
Conductivity: 10 $\mathrm{W\,m^{-1}\,K}$
Specific Heat: 480 $\mathrm{J\,kg^{-1}}$
Density: 4800 $\mathrm{kg\,m^{-3}}$
Emissivity: 0.03
Heat Flow: 30 W
Temperature Boundary Condition: None

Contact 4: BaseplateXZ to PowerBattery
Contact Conductivity: 4000 $\mathrm{W\,m^{-2}\,K}$
Contact Partner 1 mesh size: 5 mm
Contact Partner 2 mesh size: 5 mm

C.4 Model Dynamic

Base model modified by:

Simulation Type: Transient Timestep: 1 s Number of timesteps: 100 Total time: 100 s

Component 1: BaseplateXZ
Initial Temperature: 300 K
Characteristic mesh size: 30 mm
Material: Aluminum
Conductivity: 155 $\mathrm{W\,m^{-1}\,K}$
Specific Heat: 915 $\mathrm{J\,kg^{-1}}$
Density: 2700 $\mathrm{kg\,m^{-3}}$
Emissivity: 0.03
Heat Flow: Face 3, 1380 $\mathrm{W\,m^{-1}}$
Temperature Boundary Condition: None

Component 5: PayloadCommunications
Initial Temperature: 300 K
Characteristic mesh size: 30 mm
Material: Copper
Conductivity: 400 $\mathrm{W\,m^{-1}\,K}$
Specific Heat: 385 $\mathrm{J\,kg^{-1}}$

Density: 8900 $\mathrm{kg\,m^{-3}}$
Emissivity: 0.03
Heat Flow: None
Temperature Boundary Condition: Complete, 300 K

Component 6: PowerBattery
Initial Temperature: 300 K
Characteristic mesh size: 15 mm
Material: Titanium
Conductivity: 10 $\mathrm{W\,m^{-1}\,K}$
Specific Heat: 480 $\mathrm{J\,kg^{-1}}$
Density: 4800 $\mathrm{kg\,m^{-3}}$
Emissivity: 0.03
Heat Flow: None
Temperature Boundary Condition: None

Appendix D

Source Code

The source code that was written in the course of this thesis is quite long and therefore it is supplied in a separate data storage or separate file complementary to this document. The file is called Heibrok_Master_Thesis_Code.zip and it hosts the code for the thermal concept, the two FreeCAD scripts, and all VirSat apps. It is provided in regular *.txt text format.

Bibliography

[1] *Systems Engineering Handbook*. 2016. url: https://www.nasa.gov/sites/default/files/atoms/files/nasa_systems_engineering_handbook_0.pdf.

[2] Anapathur Ramesh, Shilpa M. Reddy, and Dan K. Fitzsimmons. "Airplane System Design for Reliability and Quality." In: *IEEE International Reliability Physics Symposium (IRPS)*. Burlingame. doi: 10.1109/IRPS.2018.8353564.

[3] Hossein Arsham. *Systems Simulation. The Shortest Route to Applications*. Website. Version 9th. University of Baltimore, 1995. url: https://home.ubalt.edu/ntsbarsh/simulation/sim.htm (visited on 08/18/2020).

[4] *Systems Engineering Vision 2020*. Report. Version 2.03. International Council on Systems Engineering (INCOSE), 2007.

[5] Philipp M. Fischer. "Model Based Systems Engineering." In: *INNOspace Workshop - Model Based Systems Engineering - Raumfahrt - Schiffbau - Luftfahrt*. Vortrag. Deutsches Zentrum für Luft- und Raumfahrt. Hamburg, Feb. 27, 2018. url: https://elib.dlr.de/133039/.

[6] Philipp M. Fischer, Meenakshi Deshmukh, Aaron D. Koch, et al. "Enabling a Conceptual Data Model and Workflow Integration Environment for Concurrent Launch Vehicle Analysis." In: *International Systems & Concurrent Engineering for Space Applications Conference*. Vortrag. Deutsches Zentrum für Luft- und Raumfahrt. Oct. 1, 2018–Oct. 5, 2016. url: https://elib.dlr.de/122158/.

[7] *Compasso*. url: https://www.dlr.de/gk/desktopdefault.aspx/tabid-14174/24592_read-59743/ (visited on 02/21/2021).

[8] Thilo Schuldt, Markus Oswald, Klaus Döringshoff, et al. "Optical Technologies as a Perspective for Future GNSS." In: *6th International Colloquium on Scientific and Fundamental Aspects of GNSS/Galileo*. Vortrag. Valencia, Oct. 25–27, 2017. url: https://elib.dlr.de/120155/.

[9] *Systems Engineering Vision 2025*. Presentation. 2014.

[10] SEBoK. *Transitioning Systems Engineering to a Model-based Discipline*. Oct. 30, 2020. url: https://www.sebokwiki.org/wiki/Transitioning_Systems_Engineering_to_a_Model-based_Discipline (visited on 12/03/2020).

[11] Manas Bajaj, Bjorn Cole, and Dirk Zwemer. “Architecture to Geometry – Integrating System Models with Mechanical Design.” In: *AIAA Space and Astronautics Forum and Exposition (SPACE 2016)*. Conference Paper. Sept. 12–15, 2016.

[12] Sanford Friedenthal. “What is a Model?” In: *The Guide to the Systems Engineering Body of Knowledge(SEBoK)*. Ed. by R.J. Cloutier. Dec. 2, 2020.

[13] Laura E. Hart. *Introduction To Model-Based System Engineering (MBSE) and SysML. Presented at the Delaware Valley INCOSE Chapter Meeting*. Presentation. July 30, 2015.

[14] Jonette M. Stecklein, Jim Dabney, Brandon Dick, et al. “Error Cost Escalation Through the Project Life Cycle.” In: *14th Annual International Symposium*. Conference Paper. NASA Johnson Space Center. Toulouse, June 20–24, 2004. url: https://ntrs.nasa.gov/citations/20100036670.

[15] *Model Based Systems Engineering (MBSE)*. Feb. 26, 2020. url: https://www.nasa.gov/consortium/ModelBasedSystems (visited on 12/02/2020).

[16] Philipp M. Schaefer, Philipp M. Fischer, Nico Brehm, et al. “Toward a Digital Platform for Spacecraft Manufacturing.” In: *SECESA 2018*. Vortrag. Glasgow, Sept. 26–28, 2018.

[17] David G. Gilmore, ed. *Spacecraft Thermal Control Handbook. Volume 1: Fundamental Technologies*. The Aerospace Press, 2002. isbn: 1-884989-11-X.

[18] *Spacecraft Thermal Control*. Woodhead Publishing, 2012. isbn: 978-1-84569-996-3.

[19] SAFT. *4S1P VES16 battery. Saft’s standard design for space applications*. Datasheet. 2019. url: https://www.saftbatteries.com/products-solutions/products/4s1p-ves16-battery (visited on 12/09/2020).

[20] Russell Gray. *Lisa Pathfinder PCDU Requirement Specification*. Datasheet. 2019. url: http://emits.sso.esa.int/emits-doc/ASTRIUMLIM/LISA_PCDU/S2.ASU.RS.2019.pdf (visited on 02/11/2021).

[21] Junlan Li, Shaoze Yan, and Renyu Cai. “Thermal analysis of composite solar array subjected to space heat flux.” In: *Aerospace Science and Technology* 27.1 (2013), pp. 84–94. issn: 1270-9638. doi: https://doi.org/10.1016/j.ast.2012.06.010.

[22] R. Nave. *What is Temperature?* url: http://hyperphysics.phy-astr.gsu.edu/hbase/thermo/temper.html (visited on 12/29/2020).

[23] Herbert Windisch. *Thermodynamik*. 6th. De Gruyter, 2017. isbn: 978-3-11-053356-9.

[24] John M. Wallace and Peter V. Hobbs. *Atmospheric Science. An Introductory Survey*. Elsevier LTD, 2006. isbn: 978-012-732951-2.

[25] *Thermal Conductivity*. url: http://hyperphysics.phy-astr.gsu.edu/hbase/Tables/thrcn.html (visited on 12/29/2020).

[26] M. M. Yovanovich. “Four Decades of Research on Thermal Contact, Gap, and Joint Resistance in Microelectronics.” In: *IEEE Transactions on Components and Packaging Technologies* 28 (July 2005), pp. 182–206. doi: 10.1109/TCAPT.2005.848483.

[27] *Remote Sensing - Radiation*. Sept. 17, 1999. url: https://earthobservatory.nasa.gov/features/RemoteSensing/remote_02.php (visited on 02/11/2021).

[28] R. Nave. *Blackbody Radiation*. url: http://hyperphysics.phy-astr.gsu.edu/hbase/mod6.html (visited on 01/25/2021).

[29] Gerhard Kramm and Nicole Mölders. "Planck's Blackbody Radiation Law: Presentation in Different Domains and Determination of the Related Dimensional Constants." In: *Journal of the Calcutta Mathematical Society* 5 (Jan. 2009). url: https://www.researchgate.net/publication/23780898.

[30] Rebecca Lindsey. *Climate and Earth's Energy Budget*. Jan. 14, 2009. url: https://earthobservatory.nasa.gov/features/EnergyBalance (visited on 02/21/2021).

[31] Ernst Messerschmid and Stefanos Fasoulas. *Raumfahrtsysteme*. Springer, 2009. isbn: 978-3-540-77699-4.

[32] I. Moric. "On-ground characterization of the cold atoms space clock PHARAO." Thesis. Observatoire de Paris - SYRTE, CNES, Universite Pierre et Marie Curie, 2014. url: https://www.researchgate.net/publication/281534634.

[33] Jens Grosse. "Thermal and Mechanical Design and Simulation for the first High Precision Quantum Optics Experiment on a Sounding Rocket." Universität Bremen, 2016. url: https://elib.dlr.de/111921/.

[34] Jens Grosse, Michael Elsen, Claus Braxmaier, et al. "Thermal Control System for the MAIUS Atom Inteferometer Payloads on a VSB-30 Sounding Rocket." In: *24nd ESA Symposium on European Rocket and Balloon Programmes and Related Research*. Vortrag. Deutsches Zentrum für Luft- und Raumfahrt. Essen: European Space Agency, June 16–20, 2019. url: https://elib.dlr.de/133039/.

[35] *Payload Module Thermal Control*. url: https://www.esa.int/Applications/Observing_the_Earth/Meteorological_missions/MetOp/Thermal_control2 (visited on 02/11/2021).

[36] Randall J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations*. Siam, 2007. isbn: 978-0-898716-29-0.

[37] Klaus Knothe and Heribert Wessels. *Finite Elemente*. Springer Vieweg, 2017. isbn: 978-3-662-49352-6.

[38] P. Nithiarasu, Olgierd C. Zienkiwicz, Robert L. Taylor, et al. *The Finite Element Method Volume 1*. Elsevier Butterwoth-Heinemann, 2005. isbn: 9780750664318.

[39] Marshall Bern and Paul Plassmann. *Handbook of Computational Geometry*. Elsevier Butterwoth-Heinemann, 2000. isbn: 9780444825377.

[40] Mary Kathryn Thompson and John M. Thompson. *ANSYS Mechanical APDL for Finite Element Analysis*. Butterworth-Heinemann, 2017. isbn: 978-0-12-812981-4.

[41] NAFEMS. *The Importance of Mesh Convergence*. url: https://www.nafems.org/publications/knowledge-base/the-importance-of-mesh-convergence-part-1/ (visited on 02/09/2021).

[42] Guido Dhondt. *CacluliX CrunchiX User's Manual*. Version 2.16.

[43] G. Dhondt and K. Wittig. *CalculiX A Free Software Three-Dimensional Structural Finite Element Program*. url: http://www.calculix.de/ (visited on 02/09/2021).

[44] *Introduction to Ansys Mechanical APDL*. 2020. url: https://www.ansys.com/services/training-center/structures/introduction-to-ansys-mechanical-apdl (visited on 10/29/2020).

[45] *Introduction to Abaqus Scripting*. 2020. url: https://www.3ds.com/products-services/simulia/training/course-descriptions/introduction-to-abaqus-scripting/ (visited on 10/29/2020).

[46] Volker Schaus, Philipp M. Fischer, Daniel Lüdtke, et al. "Concurrent Engineering Software Development at German Aerospace Center - Status and Outlook -." In: *SECESA 2010*. Vortrag. Deutsches Zentrum für Luft- und Raumfahrt. Oct. 13–15, 2010. url: https://elib.dlr.de/65913/.

[47] Philipp M. Fischer, Meenakshi Deshmukh, Volker Maiwald, et al. "Conceptual Data Model - A Foundation For Successful Concurrent Engineering." In: *International Systems & Concurrent Engineering for Space Applications Conference*. Vortrag. Deutsches Zentrum für Luft- und Raumfahrt. Oct. 5–7, 2016. url: https://elib.dlr.de/119970/.

[48] Philipp M. Fischer, Daniel Lüdtke, Caroline Lange, et al. "Implementing model-based system engineering for the whole lifecycle of a spacecraft." In: *CEAS Space Journal* 9.3 (2017), pp. 351–365. issn: 1868-2502. doi: 10.1007/s12567-017-0166-4.

[49] Sarah Spangelo, James Cutler, Louise Anderson, et al. "Model based systems engineering (MBSE) applied to Radio Aurora Explorer (RAX) CubeSat mission operational scenarios." In: *IEEE Conference on Aerospace*. IEEE. Big Sky, Mar. 2–9, 2013. doi: 10.1109/AERO.2013.6496894. url: https://ieeexplore.ieee.org/document/6496894.

[50] Chris Paredis. *Model-Based Systems Engineering*. Presentation. 2008.

[51] C. Bock. *Integrating Systems Modeling with Engineering Analysis*. Presentation. Sept. 23, 2014.

[52] Johannes Groß and Stefan Rudolph. "Generating Simulation Models from UML - A FireSat Example." In: *SpringSim*. Orlando, Mar. 26–29, 2012.

[53] *ESATAN Thermal Modeling Suite*. url: https://www.esatan-tms.com/products/description.php?ID=2 (visited on 02/16/2021).

[54] *Model Center - Integrate*. url: https://www.phoenix-int.com/product/modelcenter-integrate/ (visited on 02/13/2021).

[55] Brigitte Boden, Jan Flink, Robert Mischek, et al. "RCE: An Integration Environment for Engineering and Science." In: *SoftwareX* (2020). issn: 2352-7110. url: https://elib.dlr.de/131958/.

[56] *Virtual Satellite Core User Manual*. Version 4.12.1.